```
%sh
STATUS="$(service cassandra status)"

if [[ $STATUS == *"is running"* ]]; then
    echo "Cassandra is running"
else
    echo " Cassandra not running .... Starting"
    service cassandra restart > /dev/null 2>&1 &
    echo " Started"
fi
```

READY

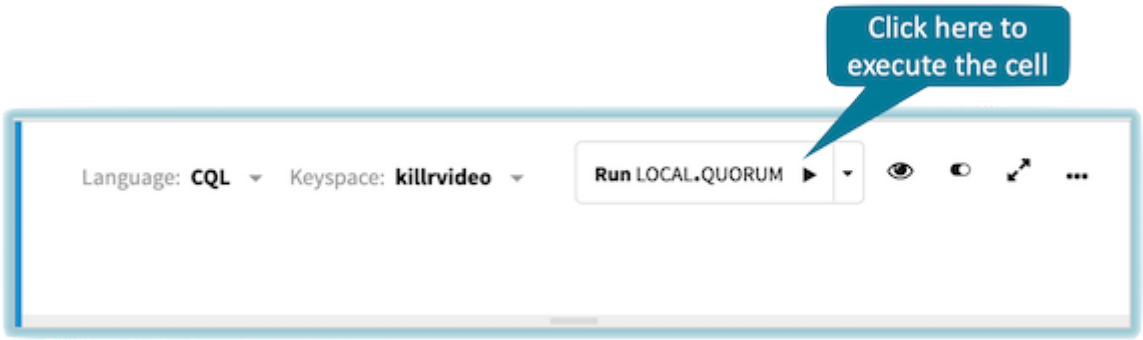
READY

Set Up the Notebook

In this section, you will do the following things:

- Execute a CQL script to initialize the KillrVideo database for this notebook

Step 1: Execute the following cell to initialize this notebook. Hover over the right-hand corner of that cell and click the *Run* button.



Note: You don't see the CQL script because the code editor is hidden, but you can still run the cell.

READY

READY

`INSERT` with Lightweight Transactions (LWTs)

In this section, you will do the following things:

- Insert a new row using an LWT

6_Advanced_Topics

- **Insert an existing row using an LWT**
- **Observe the two behaviors (and compare them to an upsert)**

Here's the story:

Imagine we want to create a new KillrVideo user. To do so, we need to create an entry in the `user_credentials` table with a specified email address. But we don't want to create a new user if there is a user already with that email address.

We could read the database to determine if the user exists already, but what if, between reading and creating the user, somebody else creates the same user. That would cause an upsert that we want to avoid.

Instead, we'll create the user with a lightweight transaction (LWT). This will make sure that Cassandra does a read-before-write as an atomic operation.

Step 1: In the following cell, Write an `INSERT` command, with an LWT, to create a user in the `user_credentials` table. Use the following table to inform your values.

Column Name	
email	'cv@fenago.
password	'3@$tC0@$tC@ss@r
userid	55555555-5555-5 5555-55555555

▶ *Need a hint? Click here.*
▶ *Just want the command? Click here.*

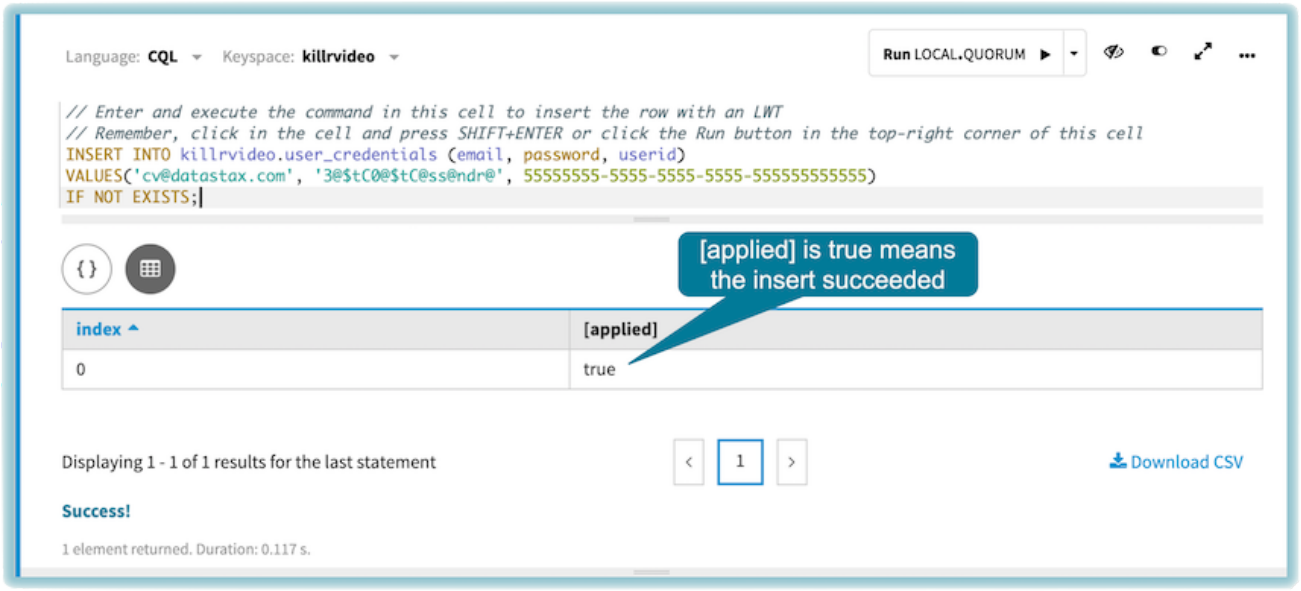
```
// Enter and execute the insert command in this cell
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

The `INSERT` returns a table with a single column named `[applied]` (with the square brackets in the name of the column).

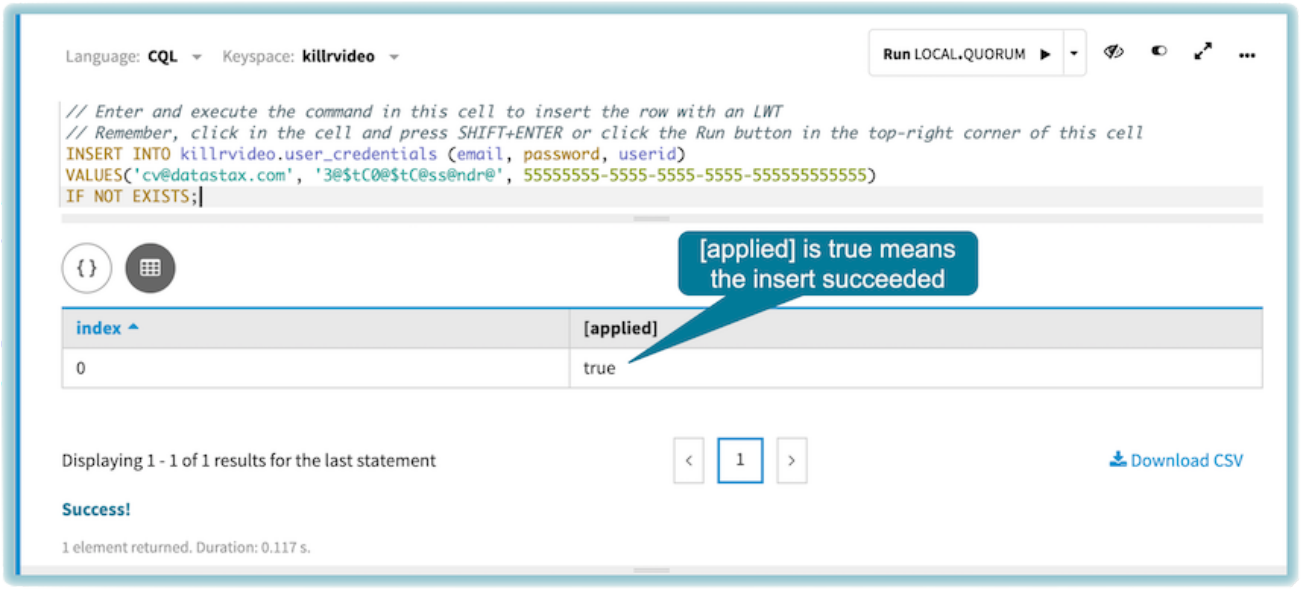
READY

6_Advanced_Topics



When the `[applied]` column has a Boolean value of `true`, it means the `INSERT` succeeded.

The `INSERT` returns a table with a single column named `[applied]` (with the square brackets in the name of the column). FINISHED



When the `[applied]` column has a Boolean value of `true`, it means the `INSERT` succeeded.

Step 2: In the next cell, run the exact same `INSERT` command you ran in Step 1.

▶ *Need a hint? Click here.*

▶ *Just want the command? Click here.*

6_Advanced_Topics

Took 0 sec. Last updated by anonymous at July 02 2020, 12:51:01 PM.

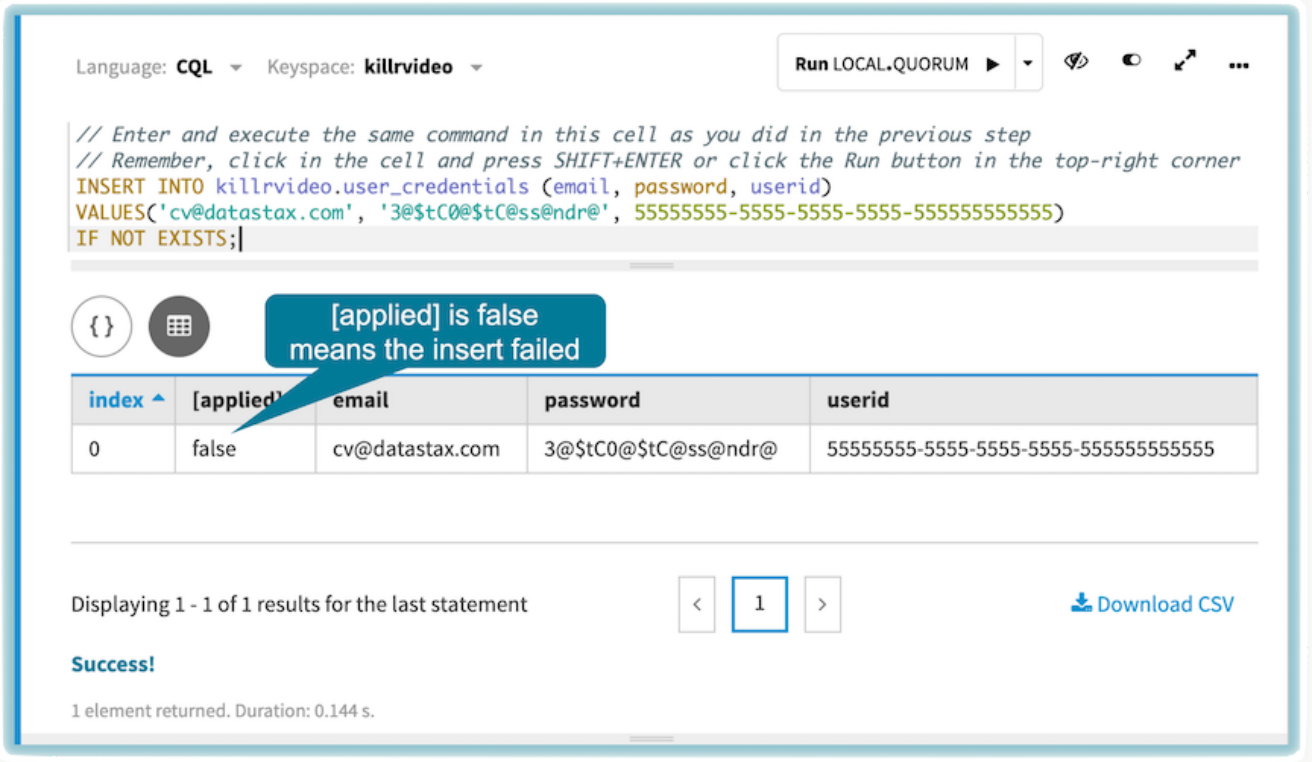
```
// Enter and execute the same command in this cell as you did in the previous step
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

Notice that this time the `INSERT` returns a row that includes the `[applied]` column along with the other columns from the table. Also note that the value of the `[applied]` column is a Boolean `false` .

READY

This means that the LWT found a record violating the condition. In this case, it's the row we previously inserted. Rather than upserting the row, the LWT returned the row it read, and noted that the write failed (in the `[applied]` column):



READY

`UPDATE` with Lightweight Transactions (LWTs)

In this section, you will do the following things:

- We'll use an `UPDATE` with an LWT that succeeds
- We'll use an `UPDATE` with an LWT that fails
- We'll explain why you care about updates and LWTs

6_Advanced_Topics

Here's the story:

Sometimes, when you want to update a record, you may run into a similar situation as you did with the insert. For example, let's say we want to update the password column, but only if the column value has not already been updated. We can use an LWT for this too.

First, let's simulate getting the current password - since that is what your code would have to do.

Step 1: In the cell that follows, write CQL to query for the user you just created

▶ *Need a hint? Click here.*

▶ *Just want the command? Click here.*

```
// Enter and execute the query to retrieve the user you created in the previous section
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

Let's imagine we want to change the password value from `3@$tC0@$tC@ss@ndr@` to `password_A`, but only if nobody has changed the password yet (we realize the security gods would hate you for using this simple password, but let's ignore them for now). We can do that with LWT.

READY

Step 2: In the cell that follows, write an `UPDATE` command to change the user's password from `'3@$tC0@$tC@ss@ndr@'` to `'Password_A'`. Use an LWT to make sure the password wasn't already changed.

▶ *Need a hint? Click here.*

▶ *Just want the command? Click here.*

```
// Enter and execute the update with LWT here
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

The LWT checks that the `password` is what we expect. Notice the resulting value in the `[applied]` column is `true`.

FINISHED

6_Advanced_Topics

What does this mean?

▶ *If you're not sure, click here.*

Now, let's simulate updating the password when it is NOT what we think it is.

For example, after we read the record to prepare for the `UPDATE`, imagine somebody else changes the password before we can. If we blindly apply our change, we may overwrite the change they applied.

Step 3: Execute an `UPDATE` in the following cell to simulate this situation (noticing that the record currently contains the password `password_A`). Set the `password` to `'password_B'` if the current password is `'3@$tC0@$tC@ss@ndr@'`.

▶ *Need a hint? Click here.*

▶ *Just want the command? Click here.*

Took 0 sec. Last updated by anonymous at July 02 2020, 12:50:02 PM.

```
// Enter and execute the update with LWT here
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

Of course, the LWT failed (as indicated by the `false` value in the `[applied]` column). Do you understand why?

READY

▶ *If you're not sure, click here.*

Just to make sure we understand, let's select the row and verify the values of its columns.

Step 4: Write a `SELECT` statement to retrieve the updated row and execute it in the following CQL cell:

▶ *Need a hint? Click here.*

▶ *Just want the command? Click here.*

```
// Enter and execute the select statement here
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

6_Advanced_Topics

READY

Batches

In this section, you will do the following things:

- **We'll explore the batch concept and understand when to use it**
- **We'll use a `BATCH` command to insert rows in each of two denormalized tables**

Note: The keyword `BATCH` has a completely different meaning in CQL than SQL. Please be careful not to confuse the two.

Here's the story:
As we have seen in Cassandra, we denormalize data so that we can build tables for fast, scalable access. By definition, denormalization requires that we insert redundant data into two or more tables. This can cause problems. Imagine two writes to two denormalized tables. If one write succeeds and the other, for reasons unknown, fails, the tables are permanently inconsistent. Batches help us solve this problem by making sure that the data gets written to both tables.

Batches sound a little like traditional transactions, and in some ways, they are. Traditional transactions will commit to both writes or, if one write fails, it rolls back the successful write. Alternatively, batches just keep trying until both writes succeed.

Two KillrVideo tables that contain denormalized data are the `users` table and the `users_credentials table` . In the previous section, we only wrote to the `user_credentials` table, so our tables are now out of sync. Let's fix that! We'll start by deleting the row we inserted in the previous sections, and then we'll add it back in to both tables using batches.

Step 1: In the following cell, delete the row with the `email` of `cv@fenago.com` from the `user_credentials` table.

▶ *Need a hint? Click here.*
▶ *Just want the command? Click here.*

```
// Enter and execute the delete command here
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

6_Advanced_Topics

Let's review the contents of both the `users` table and the `user_credentials` table.

READY

Step 2: Execute the next two cells to view the contents of both tables.

```
// This command will let you review the contents of the user_credentials table.
// This command requires a full table scan, which means it's fine on small tables like this on,
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc

SELECT * FROM killrvideo.user_credentials;
```

READY

```
// This command will let you review the contents of the users table.
// This command requires a full table scan, which means it's fine on small tables like this on,
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc

SELECT * FROM killrvideo.users;
```

READY

Note two points about these tables:

FINISHED

- **The contents of the tables are consistent - the redundant data is identical in both tables**
- **Cristina's data does not show up in either table**

Let's add a row to each table for Cristina. The following table gives you her data:

Column Name	
firstname	'Cri
lastname	'V
email	'cv@fenago.
password	'3@$tC0@$tC@ss@r
userid	55555555-5555-5 5555-55555555

Step 3: In the following cell, write and execute the CQL to insert a row into each table. Use a batch to make sure the code updates both tables.

▶ *Need a hint? Click here.*

6_Advanced_Topics

▶ *Just want the CQL? Click here.*

Took 1 sec. Last updated by anonymous at July 02 2020, 12:49:04 PM.

READY

```
// Write and execute the batch with the two inserts here
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

READY

Finally, let's review the contents of both the `users` table and the `user_credentials` table and verify the changes

Step 4: Execute the next two cells to view the contents of both tables.

READY

```
// This command will let you review the contents of the users table.
// This command requires a full table scan, which means it's fine on small tables like this on,
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
SELECT * FROM killrvideo.users;
```

READY

```
// This command will let you review the contents of the user_credentials table.
// This command requires a full table scan, which means it's fine on small tables like this on,
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
SELECT * FROM killrvideo.user_credentials;
```

READY

Review the results in the previous two cells. If you carefully compare them with results from Step 2, you should find the row in each table containing Cristina's data.

FINISHED

Congratulations!!!!

At this point you understand some of the advanced concepts of Cassandra.

You are like some kind of an Apache Cassandra PhD!

▶ *Want to feel the graduating experience? Click here.*

Took 0 sec. Last updated by anonymous at July 02 2020, 12:21:12 PM. (outdated)

6_Advanced_Topics

FINISHED

Bonus Challenge: Using TTL

If you got done early and want something to do while you wait for others, here's a bonus challenge

In this section, you will do the following things:

- **Create a table to keep track of video positions for each user**
- **We will cause entries in the new table to expire after a period of time using TTL**

Here's the pitch:
Imagine while a user is playing a video, they decide to pause it. Further, we want to keep track of where they paused so we can restart the video at that point. However, we don't want to keep track of every time every user pauses a video forever. This information is probably only useful for a limited period of time, so after that period we would like the data to just go away.
This is where Time To Live (TTL) comes in. We can set a value (in seconds) for how long a column value should exist. After that amount of time passes, Cassandra automatically removed the value from that column. Let's see how it works...

Step 1: In the next cell, write and execute the CQL to create a table named `video_positions_by_user`.

- Make the `userid` the primary key
- Include the following columns in the table

Column Name	D Ty
userid	Ul
videoid	Ul
video_position	I

▶ *Want the solution? Click here.*

Took 0 sec. Last updated by anonymous at July 02 2020, 12:21:54 PM. (outdated)

```
// Write and execute the CQL to create the video_positions_by_user table
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

6_Advanced_Topics

FINISHED

Step 2: In the following cell, insert a row into the `video_positions_by_user` table using TTL.

- **Set the TTL on the `videoid` and `video_position` values to five minutes (300 seconds) with the clause `USING TTL 300` at the end of the statement**
- **Use the values in the following table**

Column Name	Column Value
userid	**11111111-1111-1111-1111-111111111111**
videoid	**aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa**
video_position	**42**

Note: Once you execute the `INSERT` command, proceed directly to the next step. You will want to execute the next step within the TTL window. Time is ticking...

▶ *Want the solution? Click here.*

Took 0 sec. Last updated by anonymous at July 02 2020, 12:22:11 PM. (outdated)

```
// Write and execute the CQL to insert the row into video_positions_by_user table with the TTL
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

Step 3: Inspect how much time is left for the row's values.

READY

- Execute the following cell to see how much time is left on the TTL for the `videoid` value
- You may want to execute the next cell several times and watch the TTL value count down
- In fact, continue to execute this query every few seconds until the TTL expires (just to see what happens)
- Notice that Cassandra associates TTL values with column values (not the row), so the query selects the TTL for a specific column

```
// Execute this cell to see how much time is left on the TTL for the videoid value
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
SELECT videoid, TTL(videoid) FROM killrvideo.video_positions_by_user WHERE userid = 11111111-111
```

READY

6_Advanced_Topics

READY

As we mentioned, Cassandra associates the TTL with a value in the row. This implies we could set different TTLs for different columns. How might we do that? Glad you asked...

Step 4: In the following cell, insert the same row again into the `video_positions_by_user` table using TTL.

- **Again, set the TTL for to 300 seconds with the clause `USING TTL 300` at the end of the statement**
- **Here is the table again for your reference**

Column Name	Column Value
userid	11111111-1111-1111-1111-111111111111
videoid	aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
video_position	42

▶ *Want the solution? Click here.*

READY

```
// Write and execute the CQL to insert the row into video_positions_by_user table with the TTL
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc
```

READY

Step 5: Execute the following cell to reset the value of the TTL for *only* the videoid value.

Note: You may only set TTLs for values of non-primary key columns.

READY

```
// Execute this cell to see how much time is left on the TTL for the videoid value
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc

UPDATE killrvideo.video_positions_by_user
    USING TTL 300
    SET videoid = aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa
    WHERE userid = 11111111-1111-1111-1111-111111111111:
```

READY

Step 6: Execute the following cell repeatedly to watch the TTLs count down.

6_Advanced_Topics

- **Continue to execute the next cell until both TTLs expire**

- **We know, the suspense is killing you...**

```
// Execute this cell to see how much time is left on the TTL for both of the row's values
// Remember, click in the cell and press SHIFT+ENTER or click the Run button in the top-right cc

SELECT videoid, TTL(videoid), video_position, TTL(video_position) FROM killrvideo.video_positior
```

READY

READY