1.2.Creating_a_Table_with_Apache_Cassandra

August 10, 2021

1 Creating a Table with Apache Cassandra

### Walk through the basics of Apache Cassandra. Complete the following tasks:
Create a table in Apache Cassandra,
Insert rows of data,
Run a simple SQL query to validate the information. `#####` denotes where the code needs to be completed.

Note: **Do not** click the blue Preview button in the lower taskbar

Import Apache Cassandra python package

```
In [1]: import cassandra
```

1.0.1 Create a connection to the database

```
In [4]: from cassandra.cluster import Cluster
        try:
            cluster = Cluster(['127.0.0.1']) #If you have a locally installed Apache Cassandra i
            session = cluster.connect()
        except Exception as e:
            print(e)
```

1.0.2 Create a keyspace to do the work in

```
In [12]: ##  Create the keyspace
         # the name of the keybase should be lower case
         try:
             session.execute("""
             CREATE KEYSPACE IF NOT EXISTS test2
             WITH REPLICATION =
             { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }"""
         )

         except Exception as e:
             print(e)
```

1.0.3 TO-DO: Connect to the Keyspace

```
In [14]: ## Add in the keyspace you created
         try:
             session.set_keyspace('test2')
         except Exception as e:
             print(e)
```

1.0.4 Create a Song Library that contains a list of songs, including the song name, artist name, year, album it was from, and if it was a single.

```
song_title artist_name year album_name single
```

1.0.5 TO-DO: You need to create a table to be able to run the following query:

```
select * from songs WHERE year=1970 AND artist_name="The Beatles"
```

```
In [22]: ## TO-DO: Complete the query below
         query = "CREATE TABLE IF NOT EXISTS Songs "
         # query = query + "(VALUES PRIMARY KEY (year, #####))"
         query = query + "(year int, song_title text, artist_name text, album_name text, single
         try:
             session.execute(query)
         except Exception as e:
             print(e)
```

1.0.6 TO-DO: Insert the following two rows in your table

```
First Row:  "Across The Universe", "The Beatles", "1970", "False", "Let It Be"
   Second Row: "The Beatles", "Think For Yourself", "False", "1965", "Rubber Soul"
```

```
In [25]: ## Add in query and then run the insert statement
         query = "INSERT INTO Songs (year, song_title, artist_name, album_name, single) "
         query = query + " VALUES (%s, %s, %s, %s, %s)"

         try:
             session.execute(query, (1970, "Across The Universe", "The Beatles",  "Let It Be", F
         except Exception as e:
             print(e)

         try:
             session.execute(query, (1965,"Think For Yourself", "The Beatles", "Rubber Soul",  F
         except Exception as e:
             print(e)
```

1.0.7 TO-DO: Validate your data was inserted into the table.

```
In [26]: ## Complete and then run the select statement to validate the data was inserted into th
         query = 'SELECT * FROM Songs'
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         for row in rows:
             print (row.year, row.album_name, row.artist_name)
```

```
1965 Rubber Soul The Beatles
1970 Let It Be The Beatles
```

1.0.8 TO-DO: Validate the Data Model with the original query.

```
select * from songs WHERE YEAR=1970 AND artist_name="The Beatles"
```

```
In [30]: ##TO-DO: Complete the select statement to run the query
         query = "select * from songs WHERE YEAR=1970 AND artist_name='The Beatles'"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

         for row in rows:
             print (row.year, row.album_name, row.artist_name)
```

```
1970 Let It Be The Beatles
```

1.0.9 And Finally close the session and cluster connection

```
In [31]: session.shutdown()
         cluster.shutdown()
```

```
In [ ]:
```