Lesson 3 Exercise 4 Using the WHERE Clause

March 10, 2022

1 Lesson 3 Demo 4: Using the WHERE Clause

1.0.1 In this exercise we are going to walk through the basics of using the WHERE clause in Apache Cassandra.

denotes where the code needs to be completed.

We will use a python wrapper/ python driver called cassandra to run the Apache Cassandra queries. This library should be preinstalled but in the future to install this library you can run this command in a notebook to install locally: ! pip install cassandra-driver #### More documentation can be found here: https://datastax.github.io/python-driver/

Import Apache Cassandra python package

```
In [42]: import cassandra
```

1.0.2 First let's create a connection to the database

```
In [43]: from cassandra.cluster import Cluster
         try:
             cluster = Cluster(['127.0.0.1']) #If you have a locally installed Apache Cassandra
             session = cluster.connect()
         except Exception as e:
             print(e)
```

1.0.3 Let's create a keyspace to do our work in

```
In [44]: try:
             session.execute("""
             CREATE KEYSPACE IF NOT EXISTS udacity
             WITH REPLICATION =
             { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }""")
         )

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

Connect to our Keyspace. Compare this to how we had to create a new session in PostgreSQL.

```
In [45]: try:
             session.set_keyspace('udacity')
         except Exception as e:
             print(e)
```

1.0.4 Let's imagine we would like to start creating a new Music Library of albums.

1.0.5 We want to ask 4 question of our data

1. Give me every album in my music library that was released in a 1965 year

2. Give me the album that is in my music library that was released in 1965 by "The Beatles"

3. Give me all the albums released in a given year that was made in London

4. Give me the city that the album "Rubber Soul" was recorded

1.0.6 Here is our Collection of Data

1.0.7 How should we model this data? What should be our Primary Key and Partition Key? Since our data is looking for the YEAR let's start with that. From there we will add clustering columns on Artist Name and Album Name.

```
In [46]: session.execute('DROP TABLE IF EXISTS music_library')

         query = "CREATE TABLE IF NOT EXISTS music_library "
         query = query + "(year int, artist_name text, album_name text, city text, PRIMARY KEY (
         try:
             session.execute(query)
             print('Done')
         except Exception as e:
             print(e)
```

```
Done
```

1.0.8 Let's insert our data into of table

```
In [47]: query = "INSERT INTO music_library (year, artist_name, album_name, city)"
         query = query + " VALUES (%s, %s, %s, %s)"

         try:
             session.execute(query, (1970, "The Beatles", "Let it Be", "Liverpool"))
         except Exception as e:
             print(e)

         try:
```

```
        session.execute(query, (1965, "The Beatles", "Rubber Soul", "Oxford"))
    except Exception as e:
        print(e)

    try:
        session.execute(query, (1965, "The Who", "My Generation", "London"))
    except Exception as e:
        print(e)

    try:
        session.execute(query, (1966, "The Monkees", "The Monkees", "Los Angeles"))
    except Exception as e:
        print(e)

    try:
        session.execute(query, (1970, "The Carpenters", "Close To You", "San Diego"))
    except Exception as e:
        print(e)
```

1.0.9 Let's Validate our Data Model with our 4 queries.

Query 1:

```
In [48]: query = "SELECT * FROM music_library WHERE year=1965"
         try:
             rows = session.execute(query)
             for row in rows:
                 print (row.year, row.artist_name,'\n album_name:', row.album_name, row.city)

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

```
1965 The Beatles
 album_name: Rubber Soul Oxford
1965 The Who
 album_name: My Generation London
```

Let's try the 2nd query. Query 2:

```
In [49]: query = "SELECT * FROM music_library WHERE year=1965 and artist_name='The Beatles'"
         try:
             rows = session.execute(query)
         except Exception as e:
             print(e)

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

```
1965 The Beatles Rubber Soul Oxford
```

1.0.10 Let's try the 3rd query.

Query 3:

```
In [52]: query = "SELECT * FROM music_library WHERE year=1965 and city='London'"
         try:
             rows = session.execute(query)
             for row in rows:
                 print (row.year, row.artist_name, row.album_name, row.city)
         except Exception as e:
             print(e)
```

```
Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might invo
```

```
In [61]: session.execute('DROP TABLE IF EXISTS music_library')

         query = "CREATE TABLE IF NOT EXISTS music_library "
         query = query + "(year int, artist_name text, album_name text, city text, PRIMARY KEY (
         try:
             session.execute(query)
             print('Done')
         except Exception as e:
             print(e)
```

```
Done
```

```
In [62]: query = "SELECT * FROM music_library WHERE year=1965"
         try:
             rows = session.execute(query)
             for row in rows:
                 print (row.year, row.artist_name, row.album_name, row.city)
         except Exception as e:
             print(e)
```

1.0.11 Did you get an error? You can not try to access a column or a clustering column if you have not used the other defined clustering column. Let's see if we can try it a different way.

Try Query 4:

```
In [40]: query = "#####"
         try:
             rows = session.execute(query)
```

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

        for row in rows:
            print (row.city)
```

```
Done
<Error from server: code=2000 [Syntax error in CQL query] message="line 1:0 no viable alternativ
```

1.0.12 And Finally close the session and cluster connection

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

In [ ]:

In [ ]: