Project2-200574904_Satish_Kumar

September 19, 2024

0.1 Model Data using Cassandra

0.1.1 Please just submit this notebook in the Submission and make sure each cell has been executed and output is clearly displayed.

0.1.2 The aim of the project is to solve the three queries given below.

0.1.3 Introduction

There is a music streaming app called SoundCloud, that has been using their music streaming app and collecting data on songs and user activity and their aim is to analyze this data especially understanding what songs users are listening to. Currently, they are not making use of a NoSQL db and they have the data stored as a CSV file, thus its difficult for them to query the data. So our task is to create a NoSQL database for helping them with the analysis.

Import Packages

[1]:
```
!pip install cassandra-driver
```

```
Collecting cassandra-driver
  Downloading cassandra_driver-3.29.2-cp311-cp311-macosx_11_0_arm64.whl.metadata
(6.2 kB)
Collecting geomet<0.3,>=0.1 (from cassandra-driver)
  Downloading geomet-0.2.1.post1-py3-none-any.whl.metadata (1.0 kB)
Requirement already satisfied: click in /Users/anaconda3/lib/python3.11/site-
packages (from geomet<0.3,>=0.1->cassandra-driver) (8.0.4)
Requirement already satisfied: six in /Users/anaconda3/lib/python3.11/site-
packages (from geomet<0.3,>=0.1->cassandra-driver) (1.16.0)
Downloading cassandra_driver-3.29.2-cp311-cp311-macosx_11_0_arm64.whl (364 kB)
                        364.1/364.1 kB
11.8 MB/s eta 0:00:00
Downloading geomet-0.2.1.post1-py3-none-any.whl (18 kB)
Installing collected packages: geomet, cassandra-driver
Successfully installed cassandra-driver-3.29.2 geomet-0.2.1.post1

[notice] A new release of pip is
available: 24.1.1 -> 24.2
[notice] To update, run:
pip install --upgrade pip
```

[2]:
```
import pandas as pd
import numpy as np
import cassandra
import csv
from cassandra.cluster import Cluster
```

0.2 The image below is a screenshot of what the data appears like in the event_data.csv

Creating a Cluster

[7]:
```
# Task: Make a connection to the cassandra instance on your local machine(127.0.
 ↪0.1) and
# create a session to establish connection and begin executing queries

# Connect to the Cassandra cluster
cluster = Cluster()
session = cluster.connect()
```

Create & Set Keyspace

[8]:
```
# Task: Create a Keyspace and Set KEYSPACE to the keyspace specified above

# Create a keyspace
try:
    session.execute("""
    CREATE KEYSPACE IF NOT EXISTS music_library
    WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }
    """)
    print("Keyspace music_library created successfully")

except Exception as e:
    print(f"Error creating keyspace: {e}")

# Set the keyspace
session.set_keyspace('music_library')
print("\nKeyspace set")
```

```
Keyspace music_library created successfully

Keyspace set
```

0.3 List of Queries

0.3.1 1. Find the artist_name, song_title and length of song the SoundCloud app history that was heard during session_number = 338, and item_in_session_number = 4

0.3.2 2. Find the artist_name, song_title (sorted by item_in_session_number) and name(fname and lname) of the user for user_id = 10, session_number = 182

0.3.3 3. Find every name(fname and lname) of the user from the SoundCloud app history that listened to the song_title 'All Hands Against His Own'

0.3.4 Query1 Table1: How should we model this data? Think about what should be our Primary Key/Partition Key/Clustering Key

[9]:
```
## Task: Query 1: Find the artist_name, song_title and length of song the␣
 ↪SoundCloud app history
## that was heard during session_number = 338, and item_in_session_number = 4
## make use of create table command

# Create table for Query 1
try:
    session.execute("""
    CREATE TABLE IF NOT EXISTS session_songs (
        session_number INT,
        item_in_session_number INT,
        artist_name TEXT,
        song_title TEXT,
        length FLOAT,
        PRIMARY KEY (session_number, item_in_session_number)
    )
    """)
    print("Table session_songs created successfully")

except Exception as e:
    print("\n\nError occurred when creating the table",e)
```

```
Table session_songs created successfully
```

0.3.5 Let's insert our data into of table

[13]:
```
# We have provided part of the code to set up the CSV file. Please complete the␣
 ↪Apache Cassandra code below#

import csv

file_name = 'event_data.csv'

with open(file_name, encoding = 'utf8') as f:
    csv_reader = csv.reader(f)
```

```
    next(csv_reader) # skip the header in the csv file
    for row in csv_reader:
## Task: Write the INSERT statements and assign it to the query variable
        query = """
        INSERT INTO session_songs (session_number, item_in_session_number,␣
 ↪artist_name, song_title, length)
        VALUES (%s, %s, %s, %s, %s)
        """

        ## Task: Match the column in the csv file to the column in the INSERT␣
 ↪statement.
        ## e.g., if you want to INSERT gender from csv file into the database␣
 ↪you will use row[2]
        ## e.g., if you want to INSERT location from csv file into database you␣
 ↪will use row[7]
        session.execute(query, (int(row[8]), int(row[3]), row[0], row[9],␣
 ↪float(row[5])))
print("\n\nData inserted into the table successfully")
```

```
Data inserted into the table successfully
```

0.3.6 Validate our Data Model using a SELECT

[14]:
```
## Task: Make use of the SELECT statement and for loop to check if your query␣
 ↪works and display the results

# Validate data with a SELECT statement
rows = session.execute("""
SELECT artist_name, song_title, length
FROM session_songs
WHERE session_number = 338 AND item_in_session_number = 4
""")

for row in rows:
    print(f"Artist: {row.artist_name}, Song: {row.song_title}, Length: {row.
 ↪length}")
```

```
Artist: Faithless, Song: Music Matters (Mark Knight Dub), Length:
495.30731201171875
```

0.3.7 Query2 Table2: How should we model this data? Think about what should be our Primary Key/Partition Key/Clustering Key

[15]:
```
## Task: Query 2: Find the artist_name, song_title (sorted by␣
 ↪item_in_session_number) and
## name(fname and lname) of the user for user_id = 10, session_number = 182
## make use of create table command

# Create table for Query 2
try:
    session.execute("""
    CREATE TABLE IF NOT EXISTS user_session_songs (
        user_id INT,
        session_number INT,
        item_in_session_number INT,
        artist_name TEXT,
        song_title TEXT,
        first_name TEXT,
        last_name TEXT,
        PRIMARY KEY ((user_id, session_number), item_in_session_number)
    )
    """)

    print("Table user_session_songs created successfully")

except Exception as e:
    print("\n\nError occurred when creating the table",e)
```

```
Table user_session_songs created successfully
```

0.3.8 Let's insert our data into of table

[16]:
```
# We have provided part of the code to set up the CSV file. Please complete the␣
 ↪Apache Cassandra code below#
file_name = 'event_data.csv'

with open(file_name, encoding = 'utf8') as f:
    csv_reader = csv.reader(f)
    next(csv_reader) # skip the header in the csv file
    for row in csv_reader:
## Task: Write the INSERT statements and assign it to the query variable
        query = """
        INSERT INTO user_session_songs (user_id, session_number,␣
 ↪item_in_session_number, artist_name, song_title, first_name, last_name)
        VALUES (%s, %s, %s, %s, %s, %s, %s)
        """
```

```
        ## Task: Match the column in the csv file to the column in the INSERT␣
 ↪statement.
        ## e.g., if you want to INSERT gender from csv file into the database␣
 ↪you will use row[2]
        ## e.g., if you want to INSERT location from csv file into database you␣
 ↪will use row[7]
        session.execute(query, (int(row[10]), int(row[8]), int(row[3]), row[0],␣
 ↪row[9], row[1], row[4]))

print("\n\nData inserted into the table successfully")
```

```
Data inserted into the table successfully
```

0.3.9 Validate our Data Model using a SELECT

[17]:
```
## Task: Make use of the SELECT statement and for loop to check if your query␣
 ↪works and display the results

# Validate data with a SELECT statement
rows = session.execute("""
SELECT artist_name, song_title, first_name, last_name
FROM user_session_songs
WHERE user_id = 10 AND session_number = 182
ORDER BY item_in_session_number
""")

for row in rows:
    print(f"Artist: {row.artist_name}, Song: {row.song_title}, User: {row.
 ↪first_name} {row.last_name}")
```

```
Artist: Down To The Bone, Song: Keep On Keepin' On, User: Sylvie Cruz
Artist: Three Drives, Song: Greece 2000, User: Sylvie Cruz
Artist: Sebastien Tellier, Song: Kilometer, User: Sylvie Cruz
Artist: Lonnie Gordon, Song: Catch You Baby (Steve Pitron & Max Sanna Radio
Edit), User: Sylvie Cruz
```

0.3.10 Query3 Table3: How should we model this data? Think about what should be our Primary Key/Partition Key/Clustering Key

[18]:
```
## Task: Query 3: Find every name(first and lastname) of the user from the␣
 ↪SoundCloud app history that listened
## to the song_title 'All Hands Against His Own'
## make use of create table command

try:
```

```
    session.execute("""
    CREATE TABLE IF NOT EXISTS song_listeners (
        song_title TEXT,
        user_id INT,
        first_name TEXT,
        last_name TEXT,
        PRIMARY KEY (song_title, user_id)
    )
    """)
    print("Table song_listeners created successfully")

except Exception as e:
    print("\n\nError occurred when creating the table",e)
```

```
Table song_listeners created successfully
```

0.3.11 Let's insert our data into of table

[19]:
```
# We have provided part of the code to set up the CSV file. Please complete the␣
 ↪Apache Cassandra code below#
file_name = 'event_data.csv'

with open(file_name, encoding = 'utf8') as f:
    csv_reader = csv.reader(f)
    next(csv_reader) # skip the header in the csv file
    for row in csv_reader:
## Task: Write the INSERT statements and assign it to the query variable
        query = """
        INSERT INTO song_listeners (song_title, user_id, first_name, last_name)
        VALUES (%s, %s, %s, %s)
        """
        ## Task: Match the column in the csv file to the column in the INSERT␣
 ↪statement.
        ## e.g., if you want to INSERT gender from csv file into the database␣
 ↪you will use row[2]
        ## e.g., if you want to INSERT location from csv file into database you␣
 ↪will use row[7]
        session.execute(query, (row[9], int(row[10]), row[1], row[4]))

print("\n\nData inserted into the table successfully")
```

```
Data inserted into the table successfully
```

0.3.12 Validate our Data Model using a SELECT

[22]:
```
## Task: Make use of the SELECT statement and for loop to check if your query␣
 ↪works and display the results

# Validate data with a SELECT statement
rows = session.execute("""
SELECT first_name, last_name, song_title
FROM song_listeners
WHERE song_title = 'All Hands Against His Own'
""")

for row in rows:
    print(f"Song Title: {row.song_title}, User: {row.first_name} {row.
 ↪last_name}")
```

```
Song Title: All Hands Against His Own, User: Jacqueline Lynch
Song Title: All Hands Against His Own, User: Tegan Levine
Song Title: All Hands Against His Own, User: Sara Johnson
```

0.3.13 Drop the tables before closing out the sessions

[23]:
```
# Drop the tables
session.execute("DROP TABLE IF EXISTS session_songs")
session.execute("DROP TABLE IF EXISTS user_session_songs")
session.execute("DROP TABLE IF EXISTS song_listeners")
```

[23]:
```
<cassandra.cluster.ResultSet at 0x130f839d0>
```

0.3.14 Close the session and cluster connection¶

[24]:
```
# Close the session and cluster connection
session.shutdown()
cluster.shutdown()
```

[ ]:
```

```