Student Name: **Chuqi Wang** UCInetID: **chuqiw4**

1.

a) CQL Query: DESCRIBE hoofers;

b) Result:

A screen shot of a computer program

Description automatically generated

c) Answers:

The Hoofers keyspace maintain 3 copies of data, it resides in us-east1 cloud region. The read quorum size (R) and write quorum size (W) for consistent reads and writes are both 2 since read quorum = write quorum = 3/2 + 1 = 2.

2.

a) CQL CREATE Statements:

CREATE TABLE Users (

user\_id text,

email text,

joined\_date date,

nickname text,

street text,

city text,

state text,

zip text,

genres text,

PRIMARY KEY (user\_id)

);

CREATE TABLE Records (

record\_id text,

artist\_user\_id text,

title text,

genre text,

release\_date date,

PRIMARY KEY (record\_id)

);

CREATE TABLE Reviews (

review\_id text,

user\_id text,

record\_id text,

rating int,

body text,

posted\_at timestamp,

PRIMARY KEY (review\_id)

);

CREATE TABLE Sessions (

session\_id text,

user\_id text,

record\_id text,

track\_number int,

initiate\_at timestamp,

leave\_at timestamp,

music\_quality text,

device text,

remaining\_time int,

replay\_count int,

PRIMARY KEY (session\_id)

);

3.

a) PostgreSQL COPY commands:

\COPY zotmusic.Users to 'Users/chuqiwang/Desktop/UCI/CS224P/assignments/HW2/setup/users.csv' DELIMITER ',' CSV HEADER;

\COPY zotmusic.Records to 'Users/chuqiwang/Desktop/UCI/CS224P/assignments/HW2/setup/records.csv' DELIMITER ',' CSV HEADER;

\COPY zotmusic.Reviews to 'Users/chuqiwang/Desktop/UCI/CS224P/assignments/HW2/setup/reviews.csv' DELIMITER ',' CSV HEADER;

\COPY zotmusic.Sessions to 'Users/chuqiwang/Desktop/UCI/CS224P/assignments/HW2/setup/sessions.csv' DELIMITER ',' CSV HEADER;

A screenshot of a computer program

Description automatically generated

4.

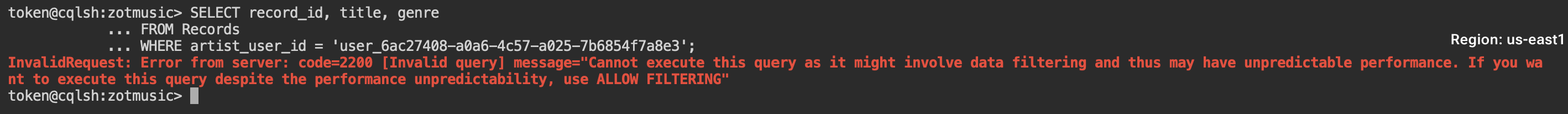
a) First CQL Query:

SELECT record\_id, title, genre

FROM Records

WHERE artist\_user\_id = 'user\_6ac27408-a0a6-4c57-a025-7b6854f7a8e3';

b) Result:



c) Modified CQL Query:

SELECT record\_id, title, genre FROM Records

WHERE artist\_user\_id = 'user\_6ac27408-a0a6-4c57-a025-7b6854f7a8e3' ALLOW FILTERING;

b) Result:

A screenshot of a computer

Description automatically generated

5.

a) CQL Create Statement:

CREATE TABLE Records\_q5 (

record\_id text,

artist\_user\_id text,

title text,

genre text,

release\_date date,

PRIMARY KEY (artist\_user\_id, record\_id)

);

b) CQL Query:

SELECT record\_id, title, genre

FROM Records\_q5

WHERE artist\_user\_id = 'user\_6ac27408-a0a6-4c57-a025-7b6854f7a8e3';

c) Result:

A screenshot of a computer

Description automatically generated

d) Explanation:

Changing the partitioning key to artist\_user\_id did altered Cassandra’s behavior because Cassandra is set to use partition key to find the location of data. In question 4, artist\_user\_id is not partition key, so Cassandra needs to scan multiple partitions to locate the data and it need allow filtering. And why we also need to include record\_id in the primary key is because PRIMARY KEY (record\_id, artist\_user\_id) ensures each record is unique for a given artist. If we don’t include record\_id in the primary key, then record\_id will not be unique, but record\_id must be unique.

6.

a) CQL Query:

SELECT record\_id, title, release\_date

FROM Records

WHERE artist\_user\_id = 'user\_bab3f848-261f-4056-a865-4f01793058a3'

ORDER BY release\_date DESC

LIMIT 5;

SELECT record\_id, title, release\_date

FROM Records\_q5

WHERE artist\_user\_id = 'user\_bab3f848-261f-4056-a865-4f01793058a3'

ORDER BY release\_date DESC

LIMIT 5;

SELECT record\_id, title, release\_date

FROM Records\_q6

WHERE artist\_user\_id = 'user\_bab3f848-261f-4056-a865-4f01793058a3'

LIMIT 5;

b) CQL CREATE Statement:

CREATE TABLE Records\_q6 (

record\_id text,

artist\_user\_id text,

title text,

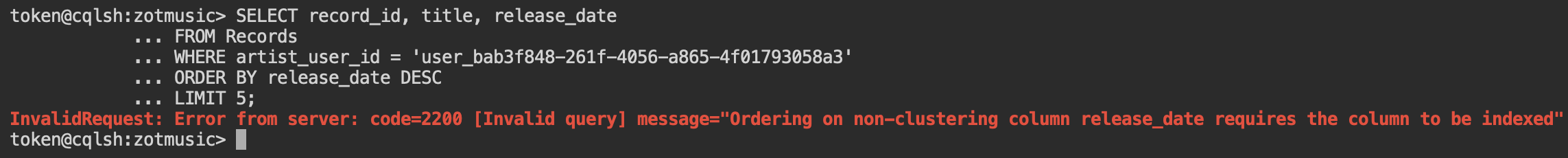
genre text,

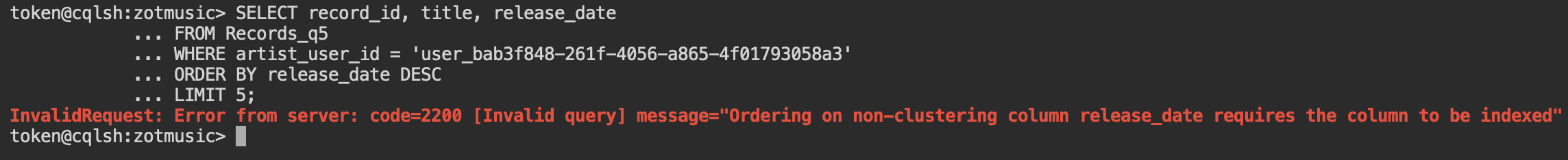
release\_date date,

PRIMARY KEY (artist\_user\_id, release\_date, record\_id)

) WITH CLUSTERING ORDER BY (release\_date DESC);

c) Results:





A screenshot of a computer program

Description automatically generated

d) Explanation:

Adding release\_date as a clustering key and specifying a clustering order by (release\_date DESC) ensures Cassandra fill in the data for each artist and sorted by their release\_date. In PRIMARY KEY (artist\_user\_id, release\_date, record\_id), release\_date is the clustering key and record\_id is the second clustering key. Including record\_id ensures each records is unique within the same partition.

7.

a) CQL Create Statement:

CREATE TABLE Reviews\_q7a (

review\_id text,

user\_id text,

record\_id text,

rating int,

body text,

posted\_at timestamp,

PRIMARY KEY (user\_id, rating, review\_id)

) WITH CLUSTERING ORDER BY (rating DESC);

b) CQL Create Statement:

CREATE TABLE Records\_q7b (

artist\_user\_id text,

record\_id text,

title text,

genre text,

release\_date date,

PRIMARY KEY (genre, record\_id)

);

c) CQL Create Statement:

CREATE TABLE Reviews\_q7c (

artist\_user\_id text,

posted\_at timestamp,

review\_id text,

record\_id text,

title text,

rating int,

PRIMARY KEY (artist\_user\_id, posted\_at, review\_id)

) WITH CLUSTERING ORDER BY (posted\_at DESC);

d) CQL Create Statement:

CREATE TABLE Sessions\_q7d (

user\_id text,

record\_id text,

initiate\_at timestamp,

leave\_at timestamp,

session\_id text,

track\_number int,

replay\_count int,

music\_quality text,

device text,

remaining\_time int,

PRIMARY KEY (user\_id, initiate\_at, session\_id)

) WITH CLUSTERING ORDER BY (initiate\_at ASC);

8.

a)

* CQL Query:

SELECT review\_id, record\_id

FROM Reviews\_q7a

WHERE user\_id = 'user\_9e48cbb4-0bf9-43fc-a578-213fae51068b' LIMIT 10;

* Result:

A screenshot of a computer screen

Description automatically generated

b)

* CQL Query:

SELECT COUNT(\*)

FROM Records\_q7b

WHERE genre = 'Folk';

* Result:

A screen shot of a computer program

Description automatically generated

c)

* CQL Query:

SELECT review\_id, record\_id, title, rating

FROM Reviews\_q7c

WHERE artist\_user\_id = 'user\_6f33f39e-7659-4673-bd80-ca11394424b0' LIMIT 10;

* Result:

A screenshot of a computer

Description automatically generated

d)

* CQL Query:

SELECT MAX(replay\_count)

FROM Sessions\_q7d

WHERE user\_id = 'user\_05f9132b-47fb-4d2b-992c-17b3c4afb2df'

AND initiate\_at >= '2024-08-01 00:00:00'

AND initiate\_at <= '2024-09-01 00:00:00';

* Result:

A screenshot of a computer

Description automatically generated

9.

* CQL INSERT statements:

INSERT INTO Records (record\_id, artist\_user\_id, title, genre, release\_date)

VALUES ('record\_d2f498f8-d7ff-4f1c-a967-7090417751f5', 'user\_38eaa9f8-e8fc-4ce4-a8ae-ffb882c1786c', 'Blue By You', 'Rock', '2024-10-07');

INSERT INTO Records\_q7b (genre, record\_id, artist\_user\_id, title, release\_date)

VALUES ('Rock', 'record\_d2f498f8-d7ff-4f1c-a967-7090417751f5', 'user\_38eaa9f8-e8fc-4ce4-a8ae-ffb882c1786c', 'Blue By You', '2024-10-07');

* Verification queries:

SELECT \*

FROM Records

WHERE record\_id = 'record\_d2f498f8-d7ff-4f1c-a967-7090417751f5';

SELECT \*

FROM Records\_q7b

WHERE record\_id = 'record\_d2f498f8-d7ff-4f1c-a967-7090417751f5'

AND genre = 'Rock';

* Result:

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

10.

Python script:

from cassandra.cluster import Cluster  
from cassandra.auth import PlainTextAuthProvider  
  
def connect\_astra():  
 cloud\_config = {  
 'secure\_connect\_bundle': '/Users/chuqiwang/Desktop/UCI/CS224P/assignments/HW2/setup/secure-connect-cs224p-fall.zip'  
 }  
 auth\_provider = PlainTextAuthProvider('SeNvcKCScXavolPBtUFOnUXX',  
 'Zk0,qmc9uITdUNs7X1sZrIt-StdX1lloZyrzxy6fFxwUhnJc84foe+lmksNRzfzd4kr\_tFSN\_RBlhaD5kCSl,Z0fZznU9OUqfpaJw3JQ,13XkMCPSDN9jrmj7JuzvLyz')  
 cluster = Cluster(cloud=cloud\_config, auth\_provider=auth\_provider)  
 session = cluster.connect('zotmusic')  
 return session  
  
  
def insert\_record(session):  
 record\_id = 'record\_632fe768-eecb-4596-9780-cc21734feec5'  
 artist\_user\_id = 'user\_b91cf915-487b-42fc-b6b8-6c17935bb755'  
 title = 'One Sour Day'  
 genre = 'R&B'  
 release\_date = '2024-10-07'  
  
 session.execute(  
 """  
 INSERT INTO Records (record\_id, artist\_user\_id, title, genre, release\_date)  
 VALUES (%s, %s, %s, %s, %s)  
 """,  
 (record\_id, artist\_user\_id, title, genre, release\_date)  
 )  
  
 session.execute(  
 """  
 INSERT INTO RecordsByGenre (genre, record\_id, artist\_user\_id, title, release\_date)  
 VALUES (%s, %s, %s, %s, %s)  
 """,  
 (genre, record\_id, artist\_user\_id, title, release\_date)  
 )  
  
def main():  
 session = connect\_astra()  
 insert\_record(session)  
 session.shutdown()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

* Result:

A screenshot of a computer

Description automatically generated

A computer screen with text

Description automatically generated