Sai Aditya Thalluri (101863023)
Naga Sai Prithvi Raj Mamidala (101834072)

CS 564 Project-III Video Sharing App in Cassandra

Since we are using Cassandra, followed query driven approach to design the tables. We had built separate tables to satisfy each query. It makes the data retrieval faster. Also, we need to minimize the tables as much as possible. Also, we **installed the Cassandra database** and implemented the tables, queries with test data.

Partition key: A partition key is a value which groups all the data needed for a query into a partition.

Clustering Key: In a partition, a unique attribute for each tuple is required which is clustering key. With a combination of partition key and clustering key data each tuple in a partition will be unique.

TABLE DESIGN:

Query 1: Provide channel name and thumbnail photo url of all channels in a category.

- For the given query we need to create a table with partitions based on channel category.

```
CREATE TABLE "CHANNELS_IN_CATEGORY" (
        "Category" VARCHAR          (Partition Key)
        ,"Channel_ID" VARCHAR       (Clustering Key)
        ,"Channel_name" TEXT
        ,"Thumbnail_photo_url" TEXT
        ,PRIMARY KEY (
                "Category"
                ,"Channel_ID"
                )
        )
```

- We have considered the attributes "Channel_name" and "Thumbnail_photo_url" into the table "CHANNELS_IN_CATEGORY". They are required for each channel according to the query.
- Considering "Category" attribute from "CHANNEL" entity as the partition key. It keeps all the channels from same category in a single partition. While retrieving the data, it will be a lot faster. If all the data, we need is in a single partition then time required to fetch data decreases.

- Partition key is used for accessing partitions, in this partition a category can have many channels under it. To access tuples in category partition separately a unique attribute is required.
- Unique attribute is considered as clustering key. Each channel under a category has a unique attribute Channel_ID, which is consider as clustering key. The channels in each Category partition will be arranged in ascending order of Channel_ID.
- To satisfy query 1, we need to fetch channel name and thumbnail photo url attributes from the table for the given category.

Query 2 & 3:

2. Provide video title, duration, thumbnail photo url, and upload time of all videos uploaded in a channel.

3. Provide video title, duration, and thumbnail photo url of all videos uploaded during last month in a channel.

- For both the queries we require same attributes of videos entity and data needs to be partitioned based on the channel. So, we create a common table for both without compromising on the efficiency.

```
CREATE TABLE "VIDEOS_IN_CHANNEL" (
        "Channel_ID" VARCHAR        (Partition Key)
        ,"Video_ID" VARCHAR
        ,"Video_title" TEXT
        ,"Duration" VARCHAR
        ,"Thumbnail_photo_url" TEXT
        ,"Upload_time" TIMESTAMP    (Clustering Key)
        ,PRIMARY KEY (
                "Channel_ID"
                ,"Upload_time"
                )
        )
```

- First, we consider the attributes of video entity "Video_title", "Duration", "Thumbnail_photo_url" and "Upload_time" as satisfy the requirement in both queries.
- Then to get all the videos under a channel, we consider "Channel_ID" of CHANNEL entity as the partition key. It keeps all the videos from same channel in a single partition which makes the data retrieval faster.
- To identify each tuple uniquely in a partition, a unique attribute is required which is clustering key for the table. For each video from a channel, we have an attribute "Upload_time" which will be always unique with respect to the channel. Because we cannot upload more than one video into the same channel at the same time. So

"Upload_time" is considered as clustering key and the videos in channel category are arranged in the ascending order of "Upload_time". Having a clustering key makes the data indexing easier as the data is arranged in the order.

- For the query 3, we need to fetch the specified attributes of videos uploaded for given "Channel_ID" in the last month. We need to do this based on the "Upload_time" of each video in the channel. Even in this case, having "Upload_time" as the clustering key makes it easier to filter the data which is arranged in its order and increases the access speed.
- For query 2, we will fetch all the videos attributes for given "Channel_ID" which will be in same partition. For query 3, we will fetch the videos attributes of specified channel with condition on the clustering key "Upload_time" by extracting the month from it and equating it to last month.

Query 4: Provide first name, last name, and subscription time of all the subscribers of a channel.

- For this query, we need to fetch the names of the subscribers along with their subscription times for the given channel.

```
CREATE TABLE "CHANNEL_SUBSCRIBERS" (
"Channel_ID" VARCHAR                (Partition Key)
,"User_ID" VARCHAR                  (Clustering Key)
,"First_name" TEXT
,"Last_name" TEXT
,"Subscription_time" TIMESTAMP
,PRIMARY KEY (
        "Channel_ID"
        ,"User_ID"
        )
)
```

- Considering the attributes "First_name", "Last_name" and "Subscription_time" from the "USER" entity as they required in the query for each user who subscribes for a channel.
- Since we need all the subscribers of a channel, we will consider the corresponding "Channel_ID" of "CHANNEL" entity as the partition key. It keeps all the subscribers of a channel in the same partition which makes the data retrieval faster.
- For identifying each tuple uniquely in a partition, a unique attribute is required which is clustering key for the table. Because multiple users may have the same and the output will be confusing when we try to get specific user from the table. So, we consider the "User_ID" attribute of each user from "USER" entity as the clustering key. The subscribers of each channel are arranged based on their "User_ID" and it also data indexing faster when searched to specific users among these.

- To satisfy the query 4, we will fetch the required attributes from the table “CHANNEL_SUBSCRIBERS” for the given “Channel_ID”.

Query 5: List all the channels created by a user along with the creation time.

- According to the given query, we need to fetch all the attributes of a channel created by the specified user along with the creation time. A user can create multiple channel at the same time.

```
CREATE TABLE “CHANNELS_CREATED” (
        “User_ID” VARCHAR                (Partition Key)
        ,“Channel_ID” VARCHAR            (Clustering Key)
        ,“Channel_name” TEXT
         ,“Description” TEXT
        ,“Category” VARCHAR
        ,“Thumbnail_photo_url” TEXT
         ,“Creation_time” TIMESTAMP
        ,PRIMARY KEY (
                “User_ID”
                 ,”Channel_ID”
         )
)
```

  - Considered the attributes “Channel_name”, “Thumbnail_photo_url”, “Description”, “Creation_time”, “Category” and “Creation_time” as they are required for each channel according to the query. As they query says to list the channels, we need to include all the attributes of CHANNEL entity in our table.
  - Since we need all the channels created by a user, considering “User_ID” from “USER” entity as the partition key. It keeps all the channels created by a user in a single partition which increases speed of data retrieval.
  - Each table must have unique attribute as a clustering key. Since partition is same for all the rows in the partition, each tuple must be unique with the combination of partition key and clustering key. Since Channel_ID is unique, consider it as a clustering key. All the channels created by the user which will be in the same partition, will be arranged in the order of Channel_ID. Adding a cluster key indexes the based on it, which helps when we need information about a specific channel created by the user.
  - To satisfy query 5, we will fetch all the columns of CHANNEL entity for the given “User_ID” along with the “Creation_time”.

Query 6: List video url of all the videos uploaded by a user.

- According to the given query, we need to fetch the video url's of all videos uploaded by a user. A user can upload multiple videos.

```
CREATE TABLE "VIDEOS_UPLOADED" (
        "User_ID" VARCHAR        (Partition Key)
        ,"Video_url" TEXT
        ,"Video_ID" VARCHAR      (Clustering Key)
        ,PRIMARY KEY (
                "User_ID"
                ,"Video_ID"
        )
)
```

  - "Video_url" from "VIDEO" entity is considered as an attribute, as we need to show up Video_url of videos uploaded by a user.
  - According to query it needs all the url's of videos that are uploaded by a user. So, from the condition described "User_ID" is considered as partition key. Data access in Cassandra is faster when we fetch data from same partition and node. As per the query, choosing the "User_ID" as partition key keeps all the required data in same partition.
  - Apart from partition key, a unique attribute as clustering key is required to identify each tuple in a partition uniquely. Video_ID is considered as clustering key. Also, making Video_ID as clustering key arranges the data with in a partition in ascending order based on clustering key. There will be case where we need URL of specific video uploaded by a specific user. In this case, clustering the data based on "Video_ID" index data according to its order and increases the speed of access.
  - For satisfying query 6, we will fetch the video url's of all videos for the given "User_ID".

Query 7: List all the comments in a video.

- For the given query, we need to fetch the user comments on a given video. A user can do multiple comments on the same video.

```
CREATE TABLE "VIDEO_COMMENTS" (
 "Video_ID" VARCHAR              (Partition Key)
,"User_ID" VARCHAR               (Clustering Key)
,"Review" TEXT
,PRIMARY KEY (
```

```
                "Video_ID"
                ,"User_ID"
                )
        )
```

- Review attribute is considered as comments by users on a video should be listed according to query.
- In this query we need to list all the comments of a specific video. So, Video_ID is considered as partition key. It keeps all the comments by users on a video within the same partition.
- It is mandatory to have a unique attribute as a clustering key. So, attribute "User_ID" from "USER" entity is considered as clustering key. The output is arranged based on User_ID, as it is unique. Also, clustering key reduces the data fetching time when we need to fetch comments of specific user on a specific video. In this case, since the tuples are arranged according the User_ID, it reduces the search time on it.
- To satisfy query 7, we will retrieve the reviews of all users on the given "Video_ID".

Query 8: List video id of all the videos liked by a user.

- For the given query, we need to get the video liked by a given user. A user can like multiple videos but can like a video only once.

```
CREATE TABLE "VIDEOS_LIKED" (
        "User_ID" VARCHAR       (Partition Key)
,       "Video_ID" VARCHAR      (Clustering Key)
        ,PRIMARY KEY (
                "User_ID"
                ,"Video_ID"
        )
)
```

- Here specifically Video_ID of videos liked by a user needed to be listed, so only User_ID and Video_ID are considered attributes.
- Here we need to obtain list of Video_ID's of all videos, which are liked by a specific user. So, as it specific by a user, User_ID is considered as a partition key. All the videos liked by a user will go into the same partition.
- Any table must have a clustering key which is a unique attribute. Since partition key is same for each tuple in a partition, clustering key will be used to identify each tuple uniquely. Here "Video_ID" attribute in "VIDEO" entity is considered as clustering key. Tuples in partition are arranged in ascending order of Video_ID.

- To satisfy query 8, we will fetch the video_id's of videos liked by given user's "User_ID".

SECTION-II:

Implementing in Cassandra

TABLE CREATION:

- After designing the table based on the queries, we download **Cassandra Datastax Edition** from below url and create these tables in Cassandra database.

  https://academy.datastax.com/planet-cassandra/cassandra

- Before creating tables, first we need to create a **KeySpace** with a class and replication factor. For this project we are creating a key space named "VIDEO_SHARING_APP".

```
CREATE KEYSPACE "VIDEO_SHARING_APP"
        WITH  REPLICATION = { 'class' : 'SimpleStrategy'
                ,'replication_factor' : 4 }
```

```
cqlsh> CREATE KEYSPACE VIDEO_SHARING_APP
   ... WITH  REPLICATION = { 'class' : 'SimpleStrategy'
   ... ,'replication_factor' : 4 }
   ... ;
cqlsh> use VIDEO_SHARING_APP;
cqlsh:video_sharing_app>
```

- Then we execute each of the tables designed above in VIDEO_SHARING_APP key space.

QUERY IMPLEMENTATION:

After the design of the tables and their insertion into the Cassandra database, now we will insert the data and implement each of the given queries and verify the outputs.

Query 1: Provide channel name and thumbnail photo url of all channels in a category.

- First let's insert the data for the table "CHANNELS_IN_CATEGORY" which is designed to satisfy this query.

```
Cassandra CQL Shell
                ... ;
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('music','C1','melody music','www.thumbnailphotos.com/c1');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('music','C2','folk music','www.thumbnailphotos.com/c2');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('music','C3','mass music','www.thumbnailphotos.com/c3');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('music','C4','background music','www.thumbnailphotos.com/c4');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('music','C5','husky music','www.thumbnailphotos.com/c5');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('dance','C6','classical dance','www.thumbnailphotos.com/c6');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('dance','C7','western dance','www.thumbnailphotos.com/c7');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('dance','C8','break dance','www.thumbnailphotos.com/c8');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('dance','C9','aerobic dance','www.thumbnailphotos.com/c9');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_IN_CATEGORY"("Category","Channel_ID","Channel_name","Thumbnail_photo_url") values('dance','C10','mass dance','www.thumbnailphotos.com/c10');
cqlsh:VIDEO_SHARING_APP>
```

- Then execute the below query to fetch the result for query 1.

```
SELECT “Channel_name”
      ,“Thumbnail_photo_url”
FROM “CHANNELS_IN_CATEGORY”
WHERE “Category” = ’music’
```

Output:

```
cqlsh:VIDEO_SHARING_APP> Select "Channel_name", "Thumbnail_photo_url" from "CHANNELS_IN_CATEGORY" where "Category"=’music’;

 Channel_name     | Thumbnail_photo_url
------------------+----------------------------
     melody music | www.thumbnailphotos.com/c1
       folk music | www.thumbnailphotos.com/c2
       mass music | www.thumbnailphotos.com/c3
 background music | www.thumbnailphotos.com/c4
      husky music | www.thumbnailphotos.com/c5

(5 rows)
```

Query 2 & 3:

- First let’s insert data into the table “VIDEOS_IN_CHANNEL” which is designed to satisfy both these queries.

```
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_IN_CHANNEL"("Channel_ID","Video_ID","Video_title", "Duration", "Thumbnail_photo_url","Upload_time") valu
es('C1','V1','video 1','1h2m','www.thumbnailphotos.com/v1','2013-02-03 04:05:00');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_IN_CHANNEL"("Channel_ID","Video_ID","Video_title", "Duration", "Thumbnail_photo_url","Upload_time") valu
es('C1','V2','video 2','2h6m','www.thumbnailphotos.com/v2','2014-02-04 05:05:01');
cqlsh:VIDEO_SHARING_APP>
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_IN_CHANNEL"("Channel_ID","Video_ID","Video_title", "Duration", "Thumbnail_photo_url","Upload_time") valu
es('C2','V3','video 3','2h6m9s','www.thumbnailphotos.com/v3','2013-02-03 04:05:02');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_IN_CHANNEL"("Channel_ID","Video_ID","Video_title", "Duration", "Thumbnail_photo_url","Upload_time") valu
es('C2','V4','video 4','6h14m24s','www.thumbnailphotos.com/v4','2014-02-04 05:05:04');
cqlsh:VIDEO_SHARING_APP>
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_IN_CHANNEL"("Channel_ID","Video_ID","Video_title", "Duration", "Thumbnail_photo_url","Upload_time") valu
es('C3','V5','video 5','3h1m2s','www.thumbnailphotos.com/v5','2013-02-03 04:05:03');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_IN_CHANNEL"("Channel_ID","Video_ID","Video_title", "Duration", "Thumbnail_photo_url","Upload_time") valu
es('C3','V6','video 6','7h49m41s','www.thumbnailphotos.com/v6','2014-02-04 05:05:05');
```

- Then execute the below query to fetch the result for query 2.

```
SELECT “Video_title”
       ,”Duration”
       ,”Thumbnail_photo_url”
       ,”Upload_time”
FROM “VIDEOS_IN_CHANNEL”
WHERE “Channel_ID” = ’C2’
```

```
cqlsh:VIDEO_SHARING_APP> Select "Video_title", "Duration", "Thumbnail_photo_url", "Upload_time" from "VIDEOS_IN_CHANNEL" where "Channel_ID"=’C2’;

 Video_title | Duration | Thumbnail_photo_url        | Upload_time
-------------+----------+----------------------------+---------------------------------
     video 3 |   2h6m9s | www.thumbnailphotos.com/v3 | 2013-02-03 11:05:02.000000+0000
     video 4 | 6h14m24s | www.thumbnailphotos.com/v4 | 2014-02-04 12:05:04.000000+0000

(2 rows)
cqlsh:VIDEO_SHARING_APP>
```

- Then execute the below query to fetch the result for query 2.

```
SELECT "Video_title"
       ,"Duration"
       ,"Thumbnail_photo_url"
FROM "VIDEOS_IN_CHANNEL"
AND “Upload_time”>’2019-04-01 00:00:00’
AND “Upload_time"<’2019-05-01 00:00:00’ ;
```

```
cqlsh:VIDEO_SHARING_APP> Select "Video_title", "Duration", "Thumbnail_photo_url" from "VIDEOS_IN_CHANNEL" where "Channel_ID"=’C3’
and "Upload_time">’2019-04-01 00:00:00’ and "Upload_time"<’2019-05-01 00:00:00’ ;

 Video_title | Duration | Thumbnail_photo_url
-------------+----------+----------------------------
     video 6 | 7h49m41s | www.thumbnailphotos.com/v6

(1 rows)
```

Query 4:

- First let’s insert data into the table “CHANNEL_SUBSCRIBERS” which is designed to satisfy the given query.

```
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C1','U1','john','cena','2013-02-04 04:05:01');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C2','U2','kane','cena','2013-02-03 04:05:02');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C3','U3','under','cena','2013-02-05 04:05:03');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C4','U4','jim','cena','2013-02-06 04:05:04');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C5','U5','kim','cena','2013-02-07 04:05:05');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C6','U6','rang','cena','2013-02-08 04:05:06');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C7','U7','rio','cena','2013-02-09 04:05:07');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNEL_SUBSCRIBERS"("Channel_ID","User_ID","First_name","Last_name","Subscription_time")
values('C8','U8','surviva','cena','2013-02-10 04:05:08');
```

- Now let's execute the below query to fetch the result for query 4.

```
SELECT "First_name"
      ,"Last_name"
      ,"Subscription_time"
FROM "CHANNEL_SUBSCRIBERS"
WHERE "Channel_ID" = 'C7';
```

```
cqlsh:VIDEO_SHARING_APP> Select "First_name", "Last_name", "Subscription_time"  from "CHANNEL_SUBSCRIBERS" where "Channel_ID"='C7';

 First_name | Last_name | Subscription_time
------------+-----------+---------------------------------
    stoinis |    kallis | 2014-02-09 11:05:17.000000+0000
        rio |      cena | 2013-02-09 11:05:07.000000+0000

(2 rows)
cqlsh:VIDEO_SHARING_APP>
```

Query 5:

- First let's insert data into to the table "CHANNELS_CREATED" which is designed to satisfy the given query.

```
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_CREATED"("User_ID","Channel_ID","Creation_time","Channel_name","Description","Category","Thumbnail_photo_url")
 values('U1','C1','2013-02-04 04:05:01','melody music','melody music channel','music','www.thumbnailphotos.com/c1');
cqlsh:VIDEO_SHARING_APP>
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_CREATED"("User_ID","Channel_ID","Creation_time","Channel_name","Description","Category","Thumbnail_photo_url")
 values('U2','C2','2013-02-04 04:05:02','folk music','folk music channel','music','www.thumbnailphotos.com/c2');
cqlsh:VIDEO_SHARING_APP>
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_CREATED"("User_ID","Channel_ID","Creation_time","Channel_name","Description","Category","Thumbnail_photo_url")
 values('U3','C3','2013-02-04 04:05:03','mass music','mass music channel','music','www.thumbnailphotos.com/c3');
cqlsh:VIDEO_SHARING_APP>
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_CREATED"("User_ID","Channel_ID","Creation_time","Channel_name","Description","Category","Thumbnail_photo_url")
 values('U4','C4','2013-02-04 04:05:04','background music','background music channel','music','www.thumbnailphotos.com/c4');
cqlsh:VIDEO_SHARING_APP>
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_CREATED"("User_ID","Channel_ID","Creation_time","Channel_name","Description","Category","Thumbnail_photo_url")
 values('U1','C5','2013-02-04 04:05:05','classical music','classical music channel','music','www.thumbnailphotos.com/c5');
cqlsh:VIDEO_SHARING_APP> insert into "CHANNELS_CREATED"("User_ID","Channel_ID","Creation_time","Channel_name","Description","Category","Thumbnail_photo_url")
 values('U1','C6','2013-02-04 04:05:06','classical dance','classical dance channel','dance','www.thumbnailphotos.com/c6');
```

- Now let's execute the below query to fetch the result for query 5.

```
SELECT “Channel_ID”
      ,“Channel_name”
      ,“Description”
      ,“Category”
      ,“Thumbnail_photo_url”
      ,“Creation_time”
FROM “CHANNELS_CREATED”
WHERE “User_ID” = ’U1’
```

```
cqlsh:VIDEO_SHARING_APP> Select "Channel_ID", "Channel_name", "Description", "Category", "Thumbnail_photo_url", "Creation_time" from "CHANNELS_CREATED"
 where "User_ID"='U1';

 Channel_ID | Channel_name    | Description            | Category | Thumbnail_photo_url         | Creation_time
------------+-----------------+------------------------+----------+-----------------------------+---------------------------------
         C1 |    melody music |    melody music channel |    music |  www.thumbnailphotos.com/c1 | 2013-02-04 11:05:01.000000+0000
        C10 |      mass dance |      mass dance channel |    dance | www.thumbnailphotos.com/c10 | 2013-02-04 11:05:10.000000+0000
         C5 | classical music | classical music channel |    music |  www.thumbnailphotos.com/c5 | 2013-02-04 11:05:05.000000+0000
         C6 | classical dance | classical dance channel |    dance |  www.thumbnailphotos.com/c6 | 2013-02-04 11:05:06.000000+0000
         C7 |   western dance |   western dance channel |    dance |  www.thumbnailphotos.com/c7 | 2013-02-04 11:05:07.000000+0000
         C8 |     break dance |     break dance channel |    dance |  www.thumbnailphotos.com/c8 | 2013-02-04 11:05:08.000000+0000
         C9 |   aerobic dance |   aerobic dance channel |    dance |  www.thumbnailphotos.com/c9 | 2013-02-04 11:05:09.000000+0000

(7 rows)
cqlsh:VIDEO_SHARING_APP>
```

Query 6:

- First let’s insert data into to the table “VIDEOS_UPLOADED” which is designed to satisfy the given query.

```
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U1','V1','www.video.com/v1');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U2','V2','www.video.com/v2');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U3','V3','www.video.com/v3');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U4','V4','www.video.com/v4');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U5','V5','www.video.com/v5');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U6','V6','www.video.com/v6');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U7','V7','www.video.com/v7');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U8','V8','www.video.com/v8');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U9','V9','www.video.com/v9');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U10','V10','www.video.com/v10');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U11','V11','www.video.com/v11');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U12','V12','www.video.com/v12');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_UPLOADED"("User_ID","Video_ID","Video_url") values('U13','V13','www.video.com/v13');
```

- Executing the below query to fetch the results for the query 6.

```
SELECT “Video_url”
FROM “VIDEOS_UPLOADED”
WHERE “User_ID” = ’U5’
```

```
cqlsh:VIDEO_SHARING_APP> Select "Video_url" from "VIDEOS_UPLOADED" where "User_ID"='U5';

 Video_url
------------------
 www.video.com/v5

(1 rows)
cqlsh:VIDEO_SHARING_APP>
```

Query 7 :

- Inserting the data into the table "VIDEO_COMMENTS" which is designed to satisfy given query.

```
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U1','V1','good');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U2','V2','good');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U3','V3','good');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U4','V4','good');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U5','V5','good');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U6','V6','good');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U7','V7','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U8','V8','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U9','V9','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U10','V10','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U11','V11','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U12','V12','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U13','V13','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U14','V14','bad');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U15','V15','excellent');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U16','V16','excellent');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U17','V17','excellent');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U18','V18','excellent');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U19','V19','excellent');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEO_COMMENTS"("User_ID","Video_ID","Review") values('U20','V20','excellent');
cqlsh:VIDEO_SHARING_APP>
```

- Executing the below query to fetch the result for query 6.

SELECT "Review"
FROM "VIDEO_COMMENTS"
WHERE "Video_ID" = 'V4'

```
cqlsh:VIDEO_SHARING_APP> Select "Review" from "VIDEO_COMMENTS" where "Video_ID"='V4';

 Review
--------
   good

(1 rows)
cqlsh:VIDEO_SHARING_APP>
```

Query 8:

- Let's insert data into the table "VIDEOS_LIKED" which is designed to satisfy the given query.

```
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U1','V1');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U2','V2');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U3','V3');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U4','V4');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U5','V7');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U6','V8');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U7','V7');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U8','V9');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U9','V9');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U10','V10');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U11','V13');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U12','V12');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U13','V13');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U14','V14');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U15','V17');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U16','V16');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U17','V17');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U18','V18');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U19','V15');
cqlsh:VIDEO_SHARING_APP> insert into "VIDEOS_LIKED"("User_ID","Video_ID") values('U20','V20');
```

- Execute the below query to fetch the result for query 8.

```
SELECT “Video_ID”
FROM “VIDEOS_LIKED”
WHERE “User_ID” = ’U9’
```

```
cqlsh:VIDEO_SHARING_APP> Select "Video_ID" from "VIDEOS_LIKED" where "User_ID"=’U9’;

 Video_ID
----------
       V2
       V9

(2 rows)
cqlsh:VIDEO_SHARING_APP>
```