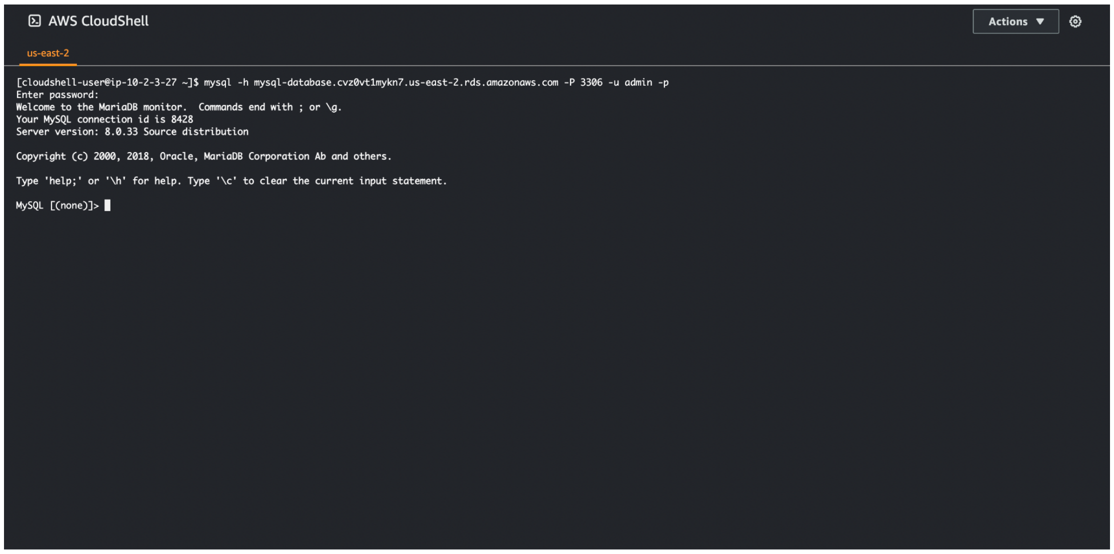
Database Implementation and Indexing

1. Platform Choice and Configuration

- **Choice of Platform**
  In stage 3, we use Amazon Web Services(AWS) rather than GCP.
- **Connections**



```
[cloudshell-user@ip-10-2-3-27 ~]$ mysql -h mysql-database.cvz0vt1mykn7.us-east-2.rds.amazonaws.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 8428
Server version: 8.0.33 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> 
```

2. Tables, DDL, and Insertion

- **Tables**
  We implemented five main tables which include core application information.
  1. Video: Contains various attributes including all the primary information about a video, one of the base tables for functionalities of advanced searching, personalized sorting, user favorites, and weekly best.
  2. Category: Contains all CategoryIds and CategoryNames, helps implement personalized sorting.
  3. Channel: Contains all ChannelIds and ChannelTitles, helps implement personalized sorting.
  4. Favorite: Includes every user's favorite videos.
  5. Tag: With all the distinct tags, one of the base tables for advanced searching and personalized sorting.
  6. TagOf: Contains videos that have tags, support advanced searching by tag.
- **DDL**
  - Video

```
-- youtube.Video definition
CREATE TABLE `Video` (
  `VideoId` varchar(255) NOT NULL,
  `Region` varchar(10) NOT NULL,
  `Title` varchar(255) NOT NULL,
  `PublishedAt` timestamp NOT NULL,
  `Likes` int NOT NULL,
  `TrendingDate` timestamp NOT NULL,
  `ViewCount` int NOT NULL,
  `ThumbnailLink` varchar(255) DEFAULT NULL,
  `LikesChange` int NOT NULL,
  `ViewCountChange` int NOT NULL,
  `ChannelId` varchar(255) DEFAULT NULL,
  `CategoryId` int DEFAULT NULL,
  PRIMARY KEY (`VideoId`,`Region`),
  FOREIGN KEY (`CategoryId`) REFERENCES `Category` (`CategoryId`) ON DELETE SET
NULL ON UPDATE CASCADE,
  FOREIGN KEY (`ChannelId`) REFERENCES `Channel` (`ChannelId`) ON DELETE SET NULL
ON UPDATE CASCADE,
  CHECK ((`Likes` >= 0)),
  CHECK ((`ViewCount` >= 0))
);
```

- Category

```
-- youtube.Category definition
CREATE TABLE `Category` (
  `CategoryId` int NOT NULL,
  `CategoryName` varchar(255) NOT NULL,
  PRIMARY KEY (`CategoryId`)
);
```

- Channel

```
-- youtube.Channel definition
CREATE TABLE `Channel` (
  `ChannelId` varchar(255) NOT NULL,
  `ChannelTitle` varchar(255) NOT NULL,
  PRIMARY KEY (`ChannelId`)
);
```

- Favorite

```
-- youtube.Favorite definition
CREATE TABLE `Favorite` (
  `VideoId` varchar(255) NOT NULL,
  `UserId` int NOT NULL,
  PRIMARY KEY (`VideoId`,`UserId`),
  FOREIGN KEY (`UserId`) REFERENCES `UserInfo` (`UserId`) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

- Tag

```
-- youtube.Tag definition
CREATE TABLE `Tag` (
  `Tag` varchar(255) NOT NULL,
  PRIMARY KEY (`Tag`)
);
```

- TagOf

```
-- youtube.TagOf definition
CREATE TABLE `TagOf` (
  `VideoId` varchar(255) NOT NULL,
  `Tag` varchar(255) NOT NULL,
  PRIMARY KEY (`VideoId`,`Tag`),
  FOREIGN KEY (`Tag`) REFERENCES `Tag` (`Tag`)
);
```

- **Insertion**
  Four of the main tables have more than 1000 insertions.

```
> select count(*) from Video
count(*)|
--------+
26214|
1 row(s) fetched.
> select count(*) from Channel
count(*)|
--------+
8628|
1 row(s) fetched.
> select count(*) from Tag
count(*)|
--------+
146093|
1 row(s) fetched.
> select count(*) from TagOf
count(*)|
--------+
345116|
1 row(s) fetched.
```

Our applications include: user favorites, advanced searching for relevant videos, top ten videos recommendations, weekly best collections globally(named must-watch videos), trending videos with personalized sorting/filtering.

3. Advanced Queries

- **Tag-based searching**

```
> select Tag from Tag where Tag in ("YOASOBI", "YOASOBI anime", "YOASOBI anime
music", "anime", "anime music", "music")
Tag |
-------+
anime |
music |
YOASOBI|
3 row(s) fetched.
> select count(distinct VideoId) from TagOf where Tag in ("anime", "music",
"YOASOBI")
count(distinct VideoId)|
-----------------------+
262|
1 row(s) fetched.
> select * from (select VideoId, sum(Importance) as Relevance from TagOf to2
natural join Factor where Tag in ("anime", "music", "YOASOBI") group by VideoId) as
t1 natural join Video where CategoryId = 10 and Region = 'JP' ORDER BY relevance
DESC limit 15 offset 0
VideoId |Relevance|Region|Title |PublishedAt |Likes|TrendingDate
|ViewCount|ThumbnailLink |LikesChange|ViewCountChange|ChannelId |CategoryId|
-----------+---------+------+-----------------------------------------------------
---------------------------+-------------------+-----+-------------------+---------
+---------------------------------------------+-----------+---------------+-------
-----------------+----------+
HPtq8YK8fDQ| 1.6603|JP |青のすみか (Live in Blue) / キタニタツヤ - Where Our Blue Is
(Live in Blue) / Tatsuya Kitani|2023-08-24 12:00:08|20233|2023-08-30 00:00:00|
405200|https://i.ytimg.com/vi/HPtq8YK8fDQ/default.jpg| 20233|
405200|UCgP3GbgbuVzAhlctGU5yuPA| 10|
-QKk9pKCufs| 0.9733|JP |【未公開映像】EXPO Behind the scenes vol.1 |NHK MUSIC EXPO
2023 | NHK |2023-09-14 03:56:38|10984|2023-09-15 00:00:00|
214928|https://i.ytimg.com/vi/-QKk9pKCufs/default.jpg| 10984|
214928|UC8T8_deSUS97DWZeKO_TL9Q| 10|
1z-_XtdtMwk| 0.9733|JP |Sexy Zone「本音と建前」(YouTube Ver.) |2023-08-23 12:00:12|
0|2023-08-30 00:00:00| 1327522|https://i.ytimg.com/vi/1z-_XtdtMwk/default.jpg| 0|
1327522|UCgXJMvOBqHk5wJFRKZfIgWQ| 10|
DxyZt6CqGe0| 0.9733|JP |Sexy Zone「本音と建前」@CDTV ライブ! ライブ! |2023-10-04 03:00:06|
0|2023-10-14 00:00:00| 609122|https://i.ytimg.com/vi/DxyZt6CqGe0/default.jpg| 0|
609122|UCgXJMvOBqHk5wJFRKZfIgWQ| 10|
mctEybOqY6s| 0.9733|JP |勇者 |2023-09-28 10:00:44|17060|2023-09-29 00:00:00|
268215|https://i.ytimg.com/vi/mctEybOqY6s/default.jpg| 17060|
268215|UCI6B8NkZKqlFWoiC_xE-hzA| 10|
epI6dTR8z8M| 0.6870|JP |アウターサイエンス / じん (Covered by 弦月藤士郎) |2023-08-24
11:00:08|21572|2023-08-30 00:00:00|
227164|https://i.ytimg.com/vi/epI6dTR8z8M/default.jpg| 21572|
227164|UCGw7lrT-rVZCWHfdG9Frcgg| 10|
fKDhZgt2LME| 0.6870|JP |Orangestar - Surges (covered by ハッピートリガー) |2023-08-31
13:30:09|40306|2023-09-01 00:00:00|
269498|https://i.ytimg.com/vi/fKDhZgt2LME/default.jpg| 40306|
269498|UC48jH1ul-6HOrcSSfoR02fQ| 10|
VJzCmHZk5LE| 0.6870|JP |アヤノの幸福理論／covered by 戌亥とこ |2023-08-21
11:00:08|39490|2023-08-30 00:00:00|
589573|https://i.ytimg.com/vi/VJzCmHZk5LE/default.jpg| 39490|
589573|UCXRlIK3Cw_TJIQC5kSJJQMg| 10|
8BU6LMzMquU| 0.3359|JP |【難易度SSS】Adoさんご本人に『唱』の歌い方を教えてもらいまSHOW。【ユニ
バーサル･スタジオ･ジャパン｢ゾンビ･デ･ダンス｣新テーマソング】|2023-09-19
09:00:07|21018|2023-09-20 00:00:00|
464667|https://i.ytimg.com/vi/8BU6LMzMquU/default.jpg| 21018|
464667|UCbVQGCsMcqC3q7OtX8iXy9Q| 10|
FpoYeL4RSm0| 0.3359|JP |モーニング娘。'23『Wake-up Call～目覚めるとき～』Promotion Edit
|2023-09-19 12:00:23|15954|2023-09-20 00:00:00|
310872|https://i.ytimg.com/vi/FpoYeL4RSm0/default.jpg| 15954|
310872|UCoKXb95K5h3sME3c9OCBaeA| 10|
mheEi3Kzhhg| 0.3359|JP |【ザ☆ピ～ス！】キングコングのお二人にダンス教えてみた ☆石田亜佑美目線
☆ |2023-10-03 09:00:32| 7840|2023-10-04 00:00:00|
179235|https://i.ytimg.com/vi/mheEi3Kzhhg/default.jpg| 7840|
179235|UCoKXb95K5h3sME3c9OCBaeA| 10|
PuE5tgJsimA| 0.3359|JP |[DAYOFF] LE SSERAFIM's DAY OFF Season3 VACANCE EP.2
|2023-08-23 10:59:58|54574|2023-08-30 00:00:00|
1074079|https://i.ytimg.com/vi/PuE5tgJsimA/default.jpg| 54574|
1074079|UCs-QBT4qkj_YiQw1ZntDO3g| 10|
Y9oD0auQrOY| 0.3359|JP |モーニング娘。'23『すっごいFEVER！』Promotion Edit |2023-10-06
12:00:29|13770|2023-10-16 00:00:00|
574585|https://i.ytimg.com/vi/Y9oD0auQrOY/default.jpg| 13770|
574585|UCoKXb95K5h3sME3c9OCBaeA| 10|
z-_vxDY1Gu0| 0.3359|JP |【超学生×四季凪アキラ】威風堂々 @歌ってみた |2023-10-12
10:00:09|46311|2023-10-16 00:00:00|
462352|https://i.ytimg.com/vi/z-_vxDY1Gu0/default.jpg| 46311|
462352|UCxIK6x6sG7Ln5vjjPYpgeAw| 10|
14 row(s) fetched.
```

- **Top ten**

```
> select Title from Video natural join Channel where CategoryId = '1' and Region =
'JP' and TrendingDate = (select max(TrendingDate) from Video) order by likes DESC
limit 10
Title |
-----------------------------------+
『劇場版 魔法少女まどか☆マギカ〈ワルプルギスの廻天〉』特報第1.1弾|
1 row(s) fetched.
```

4. Indexing Analysis

- **Tag-based searching**

We initially have the index made up with (VideoId, Region, CategoryId, ChannelId)

```
> show index from Video
Table|Non_unique|Key_name
|Seq_in_index|Column_name|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Com
ment|Index_comment|Visible|Expression|
-----+----------+---------+-----------+----------+--------+----------+--------
+------+----+----------+-------+------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK| 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK| 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
4 row(s) fetched.
```

Explain analyze:

```
> explain analyze select * from (select VideoId, sum(Importance) as Relevance from
TagOf to2 natural join Factor where Tag in ("anime", "music", "YOASOBI") group by
VideoId) as t1 natural join Video where CategoryId = 10 and Region = 'JP' ORDER BY
relevance DESC limit 5 offset 0

EXPLAIN |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+

-> Limit: 5 row(s) (cost=222300 rows=0) (actual time=1770..1770 rows=5 loops=1)
-> Nested loop inner join (cost=222300 rows=0) (actual time=1770..1770 rows=5
loops=1)
-> Sort: t1.Relevance DESC (cost=2.6..2.6 rows=0) (actual time=1770..1770 rows=7
loops=1)
-> Table scan on t1 (cost=2.5..2.5 rows=0) (actual time=1769..1770 rows=262
loops=1)
-> Materialize (cost=0..0 rows=0) (actual time=1769..1769 rows=262 loops=1)
-> Table scan on <temporary> (actual time=1768..1769 rows=262 loops=1)
-> Aggregate using temporary table (actual time=1768..1768 rows=262 loops=1)
-> Nested loop inner join (cost=261427 rows=889188) (actual time=1332..1768
rows=263 loops=1)
-> Filter: (`Factor`.Tag in ('anime','music','YOASOBI')) (cost=84994..39659
rows=352501) (actual time=1332..1767 rows=3 loops=1)
-> Table scan on Factor (cost=84994..86809 rows=144942) (actual time=1310..1644
rows=146032 loops=1)
-> Materialize (cost=84994..84994 rows=144942) (actual time=1310..1310 rows=146032
loops=1)
-> Group aggregate: count(distinct TagOf.VideoId) (cost=70500 rows=144942) (actual
time=0.0412..1116 rows=146032 loops=1)
-> Covering index skip scan for deduplication on TagOf using Tag (cost=35250
rows=352501) (actual time=0.0261..675 rows=345116 loops=1)
```

```
-> Covering index lookup on to2 using Tag (Tag=`Factor`.Tag) (cost=0.377 rows=2.52)
(actual time=0.0353..0.158 rows=87.7 loops=3)
-> Filter: (Video.CategoryId = 10) (cost=0.25 rows=0.216) (actual
time=0.0164..0.0175 rows=0.714 loops=7)
-> Single-row index lookup on Video using PRIMARY (VideoId=t1.VideoId, Region='JP')
(cost=0.25 rows=1) (actual time=0.0135..0.014 rows=0.857 loops=7)
```

First try: add (TrendingDate)

```
> create index idx_TrendingDate on Video(TrendingDate)
```

Resulting indices:

```
Table|Non_unique|Key_name |Seq_in_index|Column_name
|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Comment|Index_comment|Visibl
e|Expression|
-----+----------+----------------+------------+------------+---------+-----------+-
-------+------+----+----------+-------+-------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK | 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK | 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
Video| 1|idx_TrendingDate| 1|TrendingDate|A | 61| | | |BTREE | | |YES | |
5 row(s) fetched.
```

Explain analyze:

```
EXPLAIN |
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
------+

-> Limit: 5 row(s) (cost=222300 rows=0) (actual time=1646..1647 rows=5 loops=1)
-> Nested loop inner join (cost=222300 rows=0) (actual time=1646..1647 rows=5
loops=1)
-> Sort: t1.Relevance DESC (cost=2.6..2.6 rows=0) (actual time=1646..1646 rows=7
loops=1)
-> Table scan on t1 (cost=2.5..2.5 rows=0) (actual time=1646..1646 rows=262
loops=1)
-> Materialize (cost=0..0 rows=0) (actual time=1646..1646 rows=262 loops=1)
-> Table scan on <temporary> (actual time=1645..1645 rows=262 loops=1)
-> Aggregate using temporary table (actual time=1645..1645 rows=262 loops=1)
-> Nested loop inner join (cost=261427 rows=889188) (actual time=1254..1644
rows=263 loops=1)
-> Filter: (`Factor`.Tag in ('anime','music','YOASOBI')) (cost=84994..39659
rows=352501) (actual time=1254..1643 rows=3 loops=1)
-> Table scan on Factor (cost=84994..86809 rows=144942) (actual time=1232..1531
rows=146032 loops=1)
-> Materialize (cost=84994..84994 rows=144942) (actual time=1232..1232 rows=146032
loops=1)
-> Group aggregate: count(distinct TagOf.VideoId) (cost=70500 rows=144942) (actual
time=0.0416..1046 rows=146032 loops=1)
-> Covering index skip scan for deduplication on TagOf using Tag (cost=35250
rows=352501) (actual time=0.0265..623 rows=345116 loops=1)
-> Covering index lookup on to2 using Tag (Tag=`Factor`.Tag) (cost=0.377 rows=2.52)
(actual time=0.034..0.137 rows=87.7 loops=3)
-> Filter: (Video.CategoryId = 10) (cost=0.25 rows=0.216) (actual
time=0.0177..0.0188 rows=0.714 loops=7)
-> Single-row index lookup on Video using PRIMARY (VideoId=t1.VideoId, Region='JP')
(cost=0.25 rows=1) (actual time=0.014..0.0145 rows=0.857 loops=7)
```

Second try: add (ViewCount)

```
> create index idx_ViewCount on Video(ViewCount)
```

Resulting indice:

```
Table|Non_unique|Key_name
|Seq_in_index|Column_name|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Com
ment|Index_comment|Visible|Expression|
-----+----------+------------+-----------+----------+--------+----------+-----
---+------+----+---------+-------+------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK | 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK | 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
Video| 1|idx_ViewCount| 1|ViewCount |A | 21747| | | |BTREE | | |YES | |
5 row(s) fetched.
```

Explain analyze:

```
EXPLAIN |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+

-> Limit: 5 row(s) (cost=222300 rows=0) (actual time=1638..1638 rows=5 loops=1)
-> Nested loop inner join (cost=222300 rows=0) (actual time=1638..1638 rows=5
loops=1)
-> Sort: t1.Relevance DESC (cost=2.6..2.6 rows=0) (actual time=1638..1638 rows=7
loops=1)
```

```
-> Table scan on t1 (cost=2.5..2.5 rows=0) (actual time=1637..1637 rows=262
loops=1)
-> Materialize (cost=0..0 rows=0) (actual time=1637..1637 rows=262 loops=1)
-> Table scan on <temporary> (actual time=1636..1637 rows=262 loops=1)
-> Aggregate using temporary table (actual time=1636..1636 rows=262 loops=1)
-> Nested loop inner join (cost=261427 rows=889188) (actual time=1243..1635
rows=263 loops=1)
-> Filter: (`Factor`.Tag in ('anime','music','YOASOBI')) (cost=84994..39659
rows=352501) (actual time=1243..1635 rows=3 loops=1)
-> Table scan on Factor (cost=84994..86809 rows=144942) (actual time=1222..1523
rows=146032 loops=1)
-> Materialize (cost=84994..84994 rows=144942) (actual time=1222..1222 rows=146032
loops=1)
-> Group aggregate: count(distinct TagOf.VideoId) (cost=70500 rows=144942) (actual
time=0.046..1040 rows=146032 loops=1)
-> Covering index skip scan for deduplication on TagOf using Tag (cost=35250
rows=352501) (actual time=0.0302..620 rows=345116 loops=1)
-> Covering index lookup on to2 using Tag (Tag=`Factor`.Tag) (cost=0.377 rows=2.52)
(actual time=0.0426..0.15 rows=87.7 loops=3)
-> Filter: (Video.CategoryId = 10) (cost=0.25 rows=0.216) (actual
time=0.0171..0.0182 rows=0.714 loops=7)
-> Single-row index lookup on Video using PRIMARY (VideoId=t1.VideoId, Region='JP')
(cost=0.25 rows=1) (actual time=0.0139..0.0144 rows=0.857 loops=7)
```

Third try: add (TrendingDate, ViewCount)

```
> create index idx_TrendingDate on Video(TrendingDate)
> create index idx_ViewCount on Video(ViewCount)
```

Resulting indices:

```
Table|Non_unique|Key_name |Seq_in_index|Column_name
|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Comment|Index_comment|Visibl
e|Expression|
-----+----------+----------------+------------+------------+---------+-----------+-
-------+------+----+----------+-------+-------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK | 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK | 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
Video| 1|idx_TrendingDate| 1|TrendingDate|A | 61| | | |BTREE | | |YES | |
Video| 1|idx_ViewCount | 1|ViewCount |A | 21747| | | |BTREE | | |YES | |
6 row(s) fetched.
```

Explain analyze

```
EXPLAIN |
--------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------
----------------------------------------------------------------------------------
------+

-> Limit: 5 row(s) (cost=222300 rows=0) (actual time=1675..1675 rows=5 loops=1)
-> Nested loop inner join (cost=222300 rows=0) (actual time=1675..1675 rows=5
loops=1)
-> Sort: t1.Relevance DESC (cost=2.6..2.6 rows=0) (actual time=1675..1675 rows=7
loops=1)
-> Table scan on t1 (cost=2.5..2.5 rows=0) (actual time=1674..1675 rows=262
loops=1)
-> Materialize (cost=0..0 rows=0) (actual time=1674..1674 rows=262 loops=1)
-> Table scan on <temporary> (actual time=1673..1674 rows=262 loops=1)
-> Aggregate using temporary table (actual time=1673..1673 rows=262 loops=1)
-> Nested loop inner join (cost=261427 rows=889188) (actual time=1271..1673
rows=263 loops=1)
-> Filter: (`Factor`.Tag in ('anime','music','YOASOBI')) (cost=84994..39659
rows=352501) (actual time=1271..1672 rows=3 loops=1)
-> Table scan on Factor (cost=84994..86809 rows=144942) (actual time=1249..1556
rows=146032 loops=1)
-> Materialize (cost=84994..84994 rows=144942) (actual time=1249..1249 rows=146032
loops=1)
-> Group aggregate: count(distinct TagOf.VideoId) (cost=70500 rows=144942) (actual
time=0.0624..1065 rows=146032 loops=1)
-> Covering index skip scan for deduplication on TagOf using Tag (cost=35250
rows=352501) (actual time=0.047..627 rows=345116 loops=1)
-> Covering index lookup on to2 using Tag (Tag=`Factor`.Tag) (cost=0.377 rows=2.52)
(actual time=0.0412..0.147 rows=87.7 loops=3)
-> Filter: (Video.CategoryId = 10) (cost=0.25 rows=0.216) (actual
time=0.0168..0.018 rows=0.714 loops=7)
-> Single-row index lookup on Video using PRIMARY (VideoId=t1.VideoId, Region='JP')
(cost=0.25 rows=1) (actual time=0.0137..0.0142 rows=0.857 loops=7)
```

Report:
In the optimization process, we did not use VideoId which is used for grouping since it has already been in the default index, while all the other significant attributes inside our queries are also included in the default index.

Due to this fact, we assume that no matter what attributes from the rest can not significantly reduce actual time. Thus, we tried TrendingDate, ViewCount, and both to see if there can be obvious improvement on efficiency.

- For index TrendingDate, the result executing time is about 7% better.
- For index ViewCount, the result is nearly the same.

- For both of them as an index, we found the output leads to even less improvement than using them individually.

Based on the results, it can be concluded that the query's default index is relatively efficient enough and adding other indexes can not do much help.

- **Top ten**

We initially have the index made up with (VideoId, Region, CategoryId, ChannelId)

```
> show index from Video
Table|Non_unique|Key_name
|Seq_in_index|Column_name|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Com
ment|Index_comment|Visible|Expression|
-----+----------+----------+-----------+----------+--------+----------+--------
+------+----+----------+-------+------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK| 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK| 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
4 row(s) fetched.
```

Explain analyze:

```
> explain analyze select Title from Video natural join Channel where CategoryId =
'1' and Region = 'JP' and TrendingDate = (select max(TrendingDate) from Video)
order by likes DESC limit 10

EXPLAIN |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+

-> Limit: 10 row(s) (cost=322 rows=10) (actual time=53.6..53.6 rows=1 loops=1)
-> Nested loop inner join (cost=322 rows=910) (actual time=53.6..53.6 rows=1
loops=1)
-> Sort: Video.Likes DESC (cost=228 rows=910) (actual time=53.5..53.5 rows=1
loops=1)
-> Filter: ((Video.TrendingDate = (select #2)) and (Video.ChannelId is not null))
(cost=228 rows=910) (actual time=53.5..53.5 rows=1 loops=1)
-> Index lookup on Video using CategoryFK (CategoryId=1), with index condition:
(Video.Region = 'JP') (cost=228 rows=910) (actual time=0.226..0.656 rows=79
loops=1)
-> Select #2 (subquery in condition; run only once)
-> Aggregate: max(Video.TrendingDate) (cost=5616 rows=1) (actual time=52.7..52.7
rows=1 loops=1)
```

```
-> Table scan on Video (cost=2884 rows=27321) (actual time=0.0321..31.7 rows=26214
loops=1)
-> Single-row covering index lookup on Channel using PRIMARY
(ChannelId=Video.ChannelId) (cost=0.261 rows=1) (actual time=0.0203..0.0333 rows=1
loops=1)
```

First try: add (TrendingDate)

```
> create index idx_TrendingDate on Video(TrendingDate)
```

Resulting indices:

```
Table|Non_unique|Key_name |Seq_in_index|Column_name
|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Comment|Index_comment|Visibl
e|Expression|
-----+----------+----------------+------------+------------+---------+-----------+-
-------+------+----+---------+-------+------------+------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK | 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK | 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
Video| 1|idx_TrendingDate| 1|TrendingDate|A | 61| | | |BTREE | | |YES | |
5 row(s) fetched.
```

Explain analyze:

```
EXPLAIN |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+

-> Limit: 10 row(s) (cost=81.9 rows=10) (actual time=2.82..2.83 rows=1 loops=1)
-> Nested loop inner join (cost=81.9 rows=12.7) (actual time=2.82..2.82 rows=1
loops=1)
-> Sort: Video.Likes DESC (cost=80.3 rows=12.7) (actual time=2.8..2.8 rows=1
loops=1)
-> Filter: ((Video.TrendingDate = (select #2)) and (Video.Region = 'JP') and
(Video.CategoryId = 1) and (Video.ChannelId is not null)) (cost=80.3 rows=12.7)
(actual time=2.5..2.79 rows=1 loops=1)
-> Intersect rows sorted by row ID (cost=80.3 rows=12.7) (actual time=0.123..2.76
rows=12 loops=1)
-> Index range scan on Video using idx_TrendingDate over (TrendingDate =
'2023-10-30 00:00:00') (cost=47.5 rows=380) (actual time=0.0442..0.497 rows=380
loops=1)
-> Index range scan on Video using CategoryFK over (CategoryId = 1) (cost=28.7
rows=910) (actual time=0.0189..1.05 rows=902 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0..0 rows=1) (actual time=0.00206..0.00277
rows=1 loops=1)
-> Single-row covering index lookup on Channel using PRIMARY
(ChannelId=Video.ChannelId) (cost=0.333 rows=1) (actual time=0.00932..0.01 rows=1
loops=1)
```

Second try: add (Likes)

```
> create index idx_Likes on Video(Likes)
```

Resulting indices:

```
Table|Non_unique|Key_name
|Seq_in_index|Column_name|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Com
ment|Index_comment|Visible|Expression|
-----+----------+----------+------------+-----------+---------+-----------+--------
+------+----+----------+-------+------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK| 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK| 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
Video| 1|idx_Likes | 1|Likes |A | 17495| | | |BTREE | | |YES | |
5 row(s) fetched.
```

Explain analyze:

```
EXPLAIN |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+

-> Limit: 10 row(s) (cost=4.78 rows=0.0999) (actual time=111..171 rows=1 loops=1)
-> Nested loop inner join (cost=4.78 rows=0.0999) (actual time=111..171 rows=1
loops=1)
-> Filter: ((Video.TrendingDate = (select #2)) and (Video.Region = 'JP') and
(Video.CategoryId = 1) and (Video.ChannelId is not null)) (cost=2.5 rows=0.0999)
(actual time=111..171 rows=1 loops=1)
-> Index scan on Video using idx_Likes (reverse) (cost=2.5 rows=300) (actual
time=0.172..94 rows=26214 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Aggregate: max(Video.TrendingDate) (cost=5616 rows=1) (actual time=48.9..48.9
rows=1 loops=1)
-> Table scan on Video (cost=2884 rows=27321) (actual time=0.0298..25 rows=26214
loops=1)
-> Single-row covering index lookup on Channel using PRIMARY
(ChannelId=Video.ChannelId) (cost=0.261 rows=1) (actual time=0.0163..0.017 rows=1
loops=1)
```

Third try: add (TrendingDate, Likes)

```
> create index idx_TrendingDate on Video(TrendingDate)
> create index idx_Likes on Video(Likes)
```

Resulting indices:

```
Table|Non_unique|Key_name |Seq_in_index|Column_name
|Collation|Cardinality|Sub_part|Packed|Null|Index_type|Comment|Index_comment|Visibl
e|Expression|
-----+----------+----------------+------------+------------+---------+-----------+-
-------+------+----+----------+-------+-------------+-------+----------+
Video| 0|PRIMARY | 1|VideoId |A | 22176| | | |BTREE | | |YES | |
Video| 0|PRIMARY | 2|Region |A | 26977| | | |BTREE | | |YES | |
Video| 1|CategoryFK | 1|CategoryId |A | 15| | |YES |BTREE | | |YES | |
Video| 1|Channel_FK | 1|ChannelId |A | 8491| | |YES |BTREE | | |YES | |
Video| 1|idx_TrendingDate| 1|TrendingDate|A | 61| | | |BTREE | | |YES | |
Video| 1|idx_Likes | 1|Likes |A | 17495| | | |BTREE | | |YES | |
6 row(s) fetched.
```

Explain analyze

```
EXPLAIN |
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------+

-> Limit: 10 row(s) (cost=81.9 rows=10) (actual time=2.65..2.65 rows=1 loops=1)
-> Nested loop inner join (cost=81.9 rows=12.7) (actual time=2.64..2.65 rows=1
loops=1)
-> Sort: Video.Likes DESC (cost=80.3 rows=12.7) (actual time=2.63..2.63 rows=1
loops=1)
-> Filter: ((Video.TrendingDate = (select #2)) and (Video.Region = 'JP') and
(Video.CategoryId = 1) and (Video.ChannelId is not null)) (cost=80.3 rows=12.7)
(actual time=2.35..2.6 rows=1 loops=1)
-> Intersect rows sorted by row ID (cost=80.3 rows=12.7) (actual time=0.118..2.58
rows=12 loops=1)
-> Index range scan on Video using idx_TrendingDate over (TrendingDate =
'2023-10-30 00:00:00') (cost=47.5 rows=380) (actual time=0.0465..0.459 rows=380
loops=1)
-> Index range scan on Video using CategoryFK over (CategoryId = 1) (cost=28.7
rows=910) (actual time=0.0207..0.981 rows=902 loops=1)
-> Select #2 (subquery in condition; run only once)
-> Rows fetched before execution (cost=0..0 rows=1) (actual time=0.00193..0.00262
rows=1 loops=1)
-> Single-row covering index lookup on Channel using PRIMARY
(ChannelId=Video.ChannelId) (cost=0.333 rows=1) (actual time=0.00938..0.01 rows=1
loops=1)
```

Report:
The default index nearly covers all the attributes appearing inside the query body except TrendingDate and Likes. Thus, we tried these two and their combination.
For single TrendingDate as the additional index, the efficiency has a significant improvement of about 95%, making it such a reasonable index to stay in our top-ten query.
For single Likes, it takes even longer to execute and we literally dropped it.
For the combination of them as the index, the efficiency is approximately as good as using TrendingDate only.
As a result, we conclude that using TrendingDate as the Top-ten query's index is reasonable and helpful.