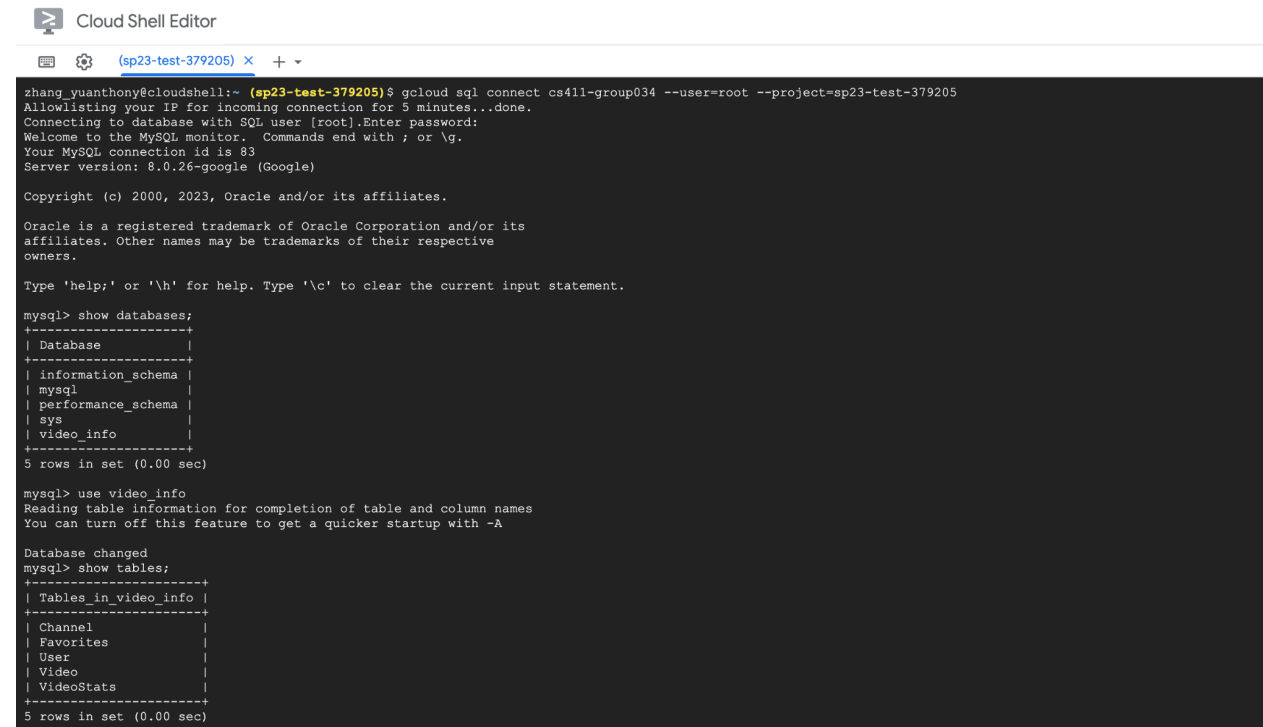


Implementation in GCP



Cloud Shell Editor

(sp23-test-379205) X + ▾

```
zhang_yuanthony@cloudshell:~ (sp23-test-379205)$ gcloud sql connect cs411-group034 --user=root --project=sp23-test-379205
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 83
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| video_info |
+-----+
5 rows in set (0.00 sec)

mysql> use video_info
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_video_info |
+-----+
| Channel |
| Favorites |
| User |
| Video |
| VideoStats |
+-----+
5 rows in set (0.00 sec)
```

DDL Commands

```
CREATE TABLE Video (
  VideoId CHAR(11) PRIMARY KEY,
  Title VARCHAR(100),
  PublishedDate VARCHAR(50),
  CategoryName VARCHAR(100),
  ChannelId CHAR(24),
  FOREIGN KEY (ChannelId) REFERENCES Channel(ChannelId)
);
```

```
CREATE TABLE VideoStats (
  VideoId CHAR(11) NOT NULL,
  TrendingDate VARCHAR(50) NOT NULL,
  ViewCount INT,
  Likes INT,
  Dislikes INT,
  PRIMARY KEY (VideoId, TrendingDate),
  FOREIGN KEY (VideoId) REFERENCES Video(VideoId)
```

```
);
/* These attributes are under VideoStats because each time a video is
trending it has different numbers for these in the dataset we are using */

CREATE TABLE Channel (
    ChannelId CHAR(24) PRIMARY KEY,
    ChannelTitle VARCHAR(255)
);

CREATE TABLE Favorites (
    UserId CHAR(6),
    VideoId CHAR(11),
    WatchListName VARCHAR(255),
    VideoAddedDate VARCHAR(50),
    PRIMARY KEY (UserId, VideoId, WatchListName)
);

CREATE TABLE User (
    UserId CHAR(6) PRIMARY KEY,
    Username VARCHAR(100),
    Password VARCHAR(255),
    Email VARCHAR(255)
);

CREATE TABLE HotVideo(
    VideoId CHAR(11) PRIMARY KEY,
    SavedCount INT
);
/* This is not one of the 4 main tables but we put it here as part of
setting up the database */
```

Table Counts

```
mysql> SELECT COUNT(*) FROM Channel;
+-----+
| COUNT(*) |
+-----+
|      1049 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Favorites;
+-----+
| COUNT(*) |
+-----+
|      1001 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM Video;
+-----+
| COUNT(*) |
+-----+
|      1654 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(*) FROM VideoStats;
+-----+
| COUNT(*) |
+-----+
|      7922 |
+-----+
1 row in set (0.01 sec)
```

Advanced Query

Query 1

This SQL query is used to count the number of favorite videos for each video category and sort the result by the number of favorite videos in descending order. As the dataset only contains 13 categories, we did not use the LIMIT clause to select the top 15 rows.

```
SELECT v.CategoryName, COUNT(*) AS num_favorites
FROM Video v JOIN Favorites f ON v.VideoId = f.VideoId
GROUP BY v.CategoryName
ORDER BY num_favorites DESC;
```

```
mysql> SELECT v.CategoryName, COUNT(*) AS num_favorites
-> FROM Video v JOIN Favorites f ON v.VideoId = f.VideoId
-> GROUP BY v.CategoryName
-> ORDER BY num_favorites DESC;
```

CategoryName	num_favorites
Music	125
Entertainment	121
Sports	68
Gaming	53
People & Blogs	53
Comedy	31
News & Politics	30
Howto & Style	29
Science & Technology	27
Film & Animation	25
Education	7
Autos & Vehicles	6
Travel & Events	4

13 rows in set (0.10 sec)

Query 2

This query is finding the 15 most popular videos with at least 10,000 views that have a high proportion of likes relative to their total reactions based on the most recent record of each video information. Since there are many records for each video on different trending dates, this approach helps to ensure that the query results only contain the most recent information.

```
SELECT v.Title, v.VideoId, v.ChannelId, n.Likes, n.Dislikes, n.ViewCount
FROM Video v JOIN (
    SELECT VideoId, ViewCount, Likes, Dislikes
    FROM VideoStats
    WHERE (VideoId, TrendingDate) IN (
        SELECT VideoId, MAX(TrendingDate)
        FROM VideoStats
        GROUP BY VideoId
    )
) AS n USING (VideoId)
WHERE n.Likes + n.Dislikes != 0 AND n.Likes / (n.Likes + n.Dislikes) > 0.95
AND n.ViewCount >= 10000
ORDER BY n.ViewCount DESC
LIMIT 15;
```

Title		VideoId	ChannelId	Likes	Dislikes	ViewCount
BTS (🇧🇴🇩🇦🇮🇱) Official MV	'Dynamite' Official MV	gdZLi9oWwZg	UCj1ZKzeVpdzPSBaWxkunda	15735533	711494	232649205
BTS (🇧🇴🇩🇦🇮🇱) Official Teaser	'Dynamite' Official Teaser	oxo8yNs9DuA	UCj1ZKzeVpdzPSBaWxkunda	6178664	158845	62496726
Ozuna x Karol G x Myke Towers - Caramelo Remix (Video Oficial)	KilyB2ma5vY	UCj1A3whiQOjSOXAXzOXBP		926272	40744	45893190
BTS (🇧🇴🇩🇦🇮🇱) Official MV (B-side)	'Dynamite' Official MV (B-side)	BVf2dGmGIWO	UCj1ZKzeVpdzPSBaWxkunda	5951286	97683	45599692
Drake - Laugh Now Cry Later (Official Music Video) ft. Lil Durk	Jcm7YDvlgnI	UCQn2uf1Sjdq65hX3zDiAl		1656221	42515	45086708
DJ Khaled ft. Drake - POPSTAR (Official Music Video - Starring Justin Bieber)	3tX7T-XtEEO	UCrPB54bgp8Sda4uduyNswIA		2109559	97497	43394819
Miley Cyrus - Midnight Sky (Official Video)		A8slnolyemTM	UC8evzfzVvAI2UVCypKTa	695129	35875	42667486
Stray Kids Back Door M/V	X-u3vTV8ScyQ	UCA06gvYLICBUStttz6zhTrzg		2077511	23985	39234949
Praw Stars: Brazil Tell - Welcome to Starr Park! Gift Shop, Colette & More!	H07pmTcorQ	UCaVwkdKzdFvCBAswpqrQ		824763	28457	32114715
I Want to Leave \$800,000 Island Keeps It	NKE0AMRspj_Y	UCXC6QdYHNE68BuQUQA		297381	35643	31801366
I Bought A Private Island	FBMlyl4mmMc	UCX6Qd3kcabNEN68BuQUQA		1801095	23693	30234876
ATEEZ(에이티즈-아이) - 'THANXX,Àø Official MV	ZKL79Ta0eiy	UCQdq-qIPeq_v2_wP_kVB9Q		605389	5071	30059975
BTS Performs Dynamite 2020 MTV VMAs	k3CDuCPUR90s	UCXAICX LdkfFYvtqTHHEOGv		2577092	58224	29442683
Better Mamba Forever Nike	C9i-WietCbK	UCUFgKR0ZHC4Rpql5VRClCA		100490	4165	29422403
Sech, Daddy Yankee, J Balvin ft. Rosalía, Farruko - ReLació'n Remix (Video Oficial)	XseJg8Vyj0	UCtc53PewI8_jYPbn07lMQo		1081016	40222	28625426
15 rows in set (0.03 sec)						

```
mysql> EXPLAIN ANALYZE SELECT v.CategoryName, COUNT(*) AS num_favorites FROM Video v JOIN Favorites f ON v.VideoId = f.VideoId GROUP BY v.CategoryName ORDER BY num_favorites DESC;
```

```
-----+-----  
| EXPLAIN  
  
-----+-----  
|  
  
-----+-----  
-> Sort: num_favorites DESC (actual time=3.469..3.470 rows=13 loops=1)  
   -> Table scan on <temporary> (actual time=0.001..0.002 rows=13 loops=1)  
       -> Aggregate using temporary table (actual time=3.437..3.439 rows=13 loops=1)  
           -> Nested loop inner join (cost=448.65 rows=992) (actual time=0.065..3.055 rows=579 loops=1)  
               -> Filter: (f.VideoId is not null) (cost=101.45 rows=992) (actual time=0.040..0.386 rows=1001 loops=1)  
                   -> Table scan on f (cost=101.45 rows=992) (actual time=0.039..0.304 rows=1001 loops=1)  
                       -> Single-row index lookup on v using PRIMARY (VideoId=f.VideoId) (cost=0.25 rows=1) (actual time=0.002..0.003 rows=1 loops=1001)
```

```
-----+-----  
1 row in set (0.00 sec)
```

```
mysql> EXPLAIN ANALYZE SELECT v.categoryName, COUNT(*) AS num_favorites FROM Video v JOIN Favorites f ON v.VideoId = f.VideoId GROUP BY v.categoryName ORDER BY num_favorites DESC;
```

```
+-----+
|
+-----+
| EXPLAIN
+-----+
|
+-----+
|
+-----+
|
+-----+
| -> Sort: num_favorites DESC (actual time=3.374..3.375 rows=13 loops=1)
|   -> Table scan on temporary<> (actual time=0.001..0.002 rows=13 loops=1)
|     -> Aggregate using temporary table (actual time=3.352..3.355 rows=13 loops=1)
|       -> Nested loop inner join (cost=448.65 rows=992) (actual time=0.068..2.935 rows=579 loops=1)
|         -> Filter: (f.VideoId is not null) (cost=101.45 rows=992) (actual time=0.041..0.369 rows=1001 loops=1)
|           -> Table scan on f (cost=101.45 rows=992) (actual time=0.040..0.292 rows=1001 loops=1)
|             -> Single-row index lookup on v using PRIMARY (VideoId=f.VideoId) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=1001)
|
+-----+
|
+-----+
| 1 row in set (0.01 sec)
```

Index Design 2 - CREATE INDEX add videoid idx ON Video(VideoId)

After using the index, the query plan remained the same, but the overall query performance improved a little, as evidenced by the reduced actual time for the "Aggregate using temporary table".

```
mysql> CREATE INDEX add_videoId_idx ON Video(VideoId);
Query OK, 0 rows affected (0.05 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> EXPLAIN ANALYZE SELECT v.CategoryName, COUNT(*) AS num_favorites FROM Video v JOIN Favorites f ON v.VideoId = f.VideoId GROUP BY v.CategoryName ORDER BY num_favorites DESC;

+-----+
| EXPLAIN |
+-----+
|         |
+-----+
|         |
+-----+
| -> Sort: num_favorites DESC (actual time=3.412..3.412 rows=13 loops=1) |
|   -> Table scan on <temporary> (actual time=0.001..0.002 rows=13 loops=1) |
|     -> Aggregate using temporary table (actual time=3.375..3.377 rows=13 loops=1) |
|       -> Nested loop inner join (cost=448.65 rows=992) (actual time=0.068..2.985 rows=579 loops=1) |
|         -> Filter: (f.VideoId is not null) (cost=101.45 rows=992) (actual time=0.041..0.382 rows=1001 loops=1) |
|           -> Table scan on f (cost=0.01..0.45 rows=992) (actual time=0.039..0.392 rows=1001 loops=1) |
|             -> Single-row index lookup on v using PRIMARY (VideoId=f.VideoId) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=1001) |
|         | |
|       | |
|     | |
|   | |
| |
+-----+
1 row in set (0.01 sec)
```

Index Design 3 - CREATE INDEX add_videoid_idx ON Favorites(VideoId)

After using the index, the query was able to utilize an index scan on the table being joined, then the query was able to use index scans to quickly locate the relevant data rows, which reduced the number of times the table needed to be scanned. This resulted in an improvement in query performance, as evidenced by the decrease in actual time from **3.469** to **2.827** milliseconds

```
mysql> CREATE INDEX add_videcid idx_on Favorites(Videoid);  
Query OK, 0 rows affected (0.05 sec)  
Records: 0 Duplicates: 0 Warnings: 0  
  
mysql> EXPLAIN ANALYZE SELECT v.CategoryName, COUNT(*) AS num_favorites FROM Video v JOIN Favorites f ON v.Videoid = f.Videoid GROUP BY v.CategoryName ORDER BY num_favorites DESC;  
+-----+  
| EXPLAIN |  
+-----+  
  
+-----+  
|               |  
+-----+  
  
+-----+  
-> Sort: num favorites DESC (actual time=2.827..2.827 rows=13 loops=1)  
-> Table scan on <temporary> (actual time=0.001..0.002 rows=13 loops=1)  
-> Aggregate using temporary table (actual time=2.804..2.806 rows=13 loops=1)  
-> Nested loop inner join (cost=448.45 rows=992) (actual time=0.057..2.452 rows=579 loops=1)  
-> Filter: (f.Videoid is not null) (cost=101.45 rows=992) (actual time=0.037..0.331 rows=1001 loops=1)  
-> Index scan on f using add_videcid_idx (cost=101.45 rows=992) (actual time=0.036..0.287 rows=1001 loops=1)  
-> Single-row index lookup on v using PRIMARY (Videoid=r.Videoid) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=1001)  
|  
+-----+  
  
1 row in set (0.00 sec)
```

Query 2

Initial result of EXPLAIN ANALYZE without using indexing

[illegible]

Index Design 1 - CREATE INDEX add _viewcount_idx ON VideoStats(ViewCount)

We created an index on column **ViewCount** of table **VideoStats**. The original query performs a table scan on VideoStats and uses a nested loop join to join it with the Video table. After adding an index on the ViewCount column, the optimizer changes the query plan to use the index range scan instead of a table scan, which significantly reduces the number of rows being scanned. This results in a much lower cost and faster execution time. Additionally, the filter condition is optimized to use the index on ViewCount and is applied earlier in the query execution, resulting in a further reduction in the number of rows being processed.

[illegible]

Index Design 2 - CREATE INDEX add likes_idx ON VideoStats(Likes)

We then created an index on column **Likes** of table **VideoStats**. From the EXPLAIN ANALYZE output, we can see that adding the **add_likes_idx** did not have any noticeable impact on the cost or the actual execution time of the query. This may be because the query mostly looks for the videos and then checks their like to dislike ratio, the likes aren't what we are searching through though so adding an index on them doesn't make a difference.

[illegible]

Index Design 3 - CREATE INDEX add dislikes_idx ON VideoStats(Dislikes)

After that, we tried to use **add_dislikes_idx**. Comparing the two EXPLAIN ANALYZE results for the query after and before adding this index on VideoStats.VideoId, there is no noticeable change in the execution plan, and the cost and actual time for each step remain almost the same. This is for the same reasons as Index Design 2.

