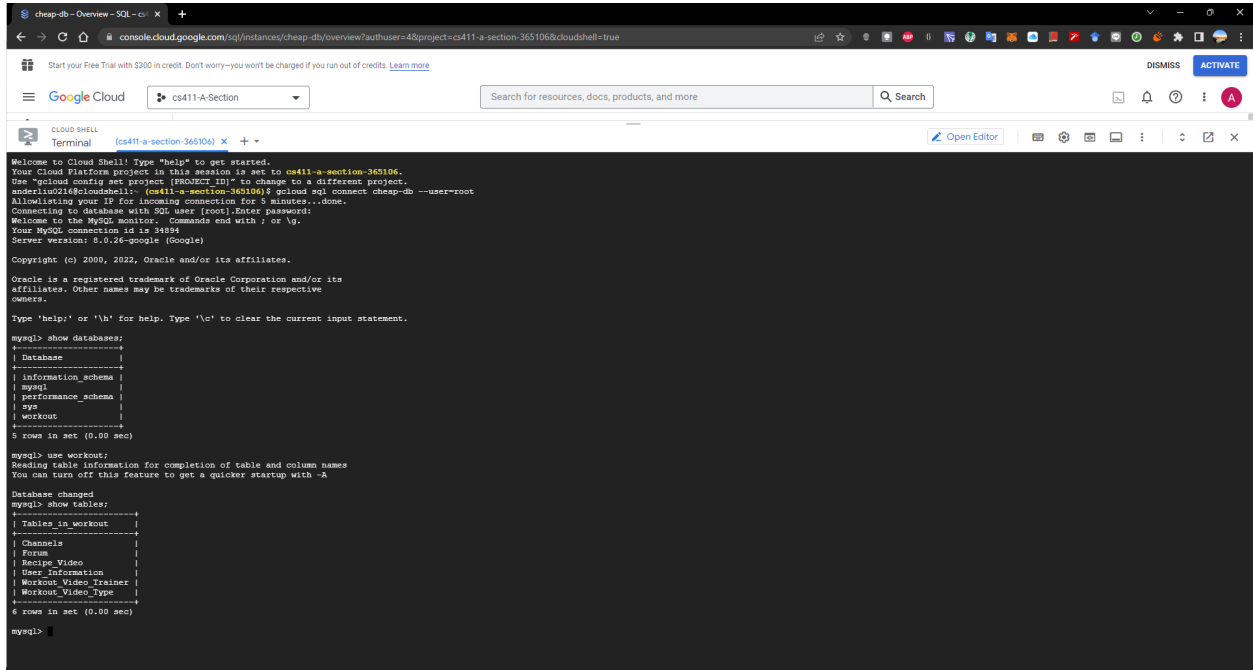


Project Stage 3

1. Database implementation: a. MySQL@GCP



```
cloudshell:~$ gcloud sql connect cheap-db --user=root
Connecting to database with SQL user [root]. Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 34894
Server version: 8.0.26-google (Google)

Copyright (c) 2000, 2022, 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 |
| workout |
+-----+
5 rows in set (0.00 sec)

mysql> use workout;
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 workout |
+-----+
| Channels |
| Forum |
| Recipe_Video |
| User_Information |
| Workout_Video_Trainer |
| Workout_Video_Type |
+-----+
6 rows in set (0.00 sec)

mysql>
```

b. Data Definition Language (DDL)

```
CREATE TABLE Channels(
    channel_id CHAR(24) NOT NULL,
    channel_title VARCHAR(255) NOT NULL,
    subscriberCount INT NOT NULL,
    channel_viewCount BIGINT NOT NULL,
    videoCount INT NOT NULL,
    PRIMARY KEY (channel_id));
```

```
CREATE TABLE Workout_Video_Trainer(
    video_id CHAR(11) NOT NULL,
    channel_id CHAR(24) NOT NULL,
    publish_date CHAR(20) NOT NULL,
    video_title VARCHAR(255) NOT NULL,
    duration VARCHAR(255) NOT NULL,
    video_viewCount INT NOT NULL,
    likeCount INT NOT NULL,
    trainer VARCHAR(255),
    PRIMARY KEY (video_id, channel_id),
```

```
FOREIGN KEY(channel_id) REFERENCES Channels(channel_id) ON DELETE  
CASCADE);
```

```
CREATE TABLE Workout_Video_Type(  
    video_id CHAR(11) NOT NULL,  
    channel_id CHAR(24) NOT NULL,  
    publish_date CHAR(20) NOT NULL,  
    video_title VARCHAR(255) NOT NULL,  
    duration VARCHAR(255) NOT NULL,  
    video_viewCount INT NOT NULL,  
    likeCount INT NOT NULL,  
    workout_type VARCHAR(255) NOT NULL,  
    PRIMARY KEY (video_id, channel_id),  
    FOREIGN KEY (channel_id) REFERENCES Channels(channel_id) ON DELETE  
CASCADE);
```

```
CREATE TABLE Recipe_Video(  
    video_id CHAR(11) NOT NULL,  
    channel_id CHAR(24) NOT NULL,  
    publish_date CHAR(20) NOT NULL,  
    video_title VARCHAR(255) NOT NULL,  
    duration VARCHAR(255) NOT NULL,  
    video_viewCount INT NOT NULL,  
    likeCount INT NOT NULL,  
    PRIMARY KEY (video_id, channel_id),  
    FOREIGN KEY(channel_id) REFERENCES Channels(channel_id) ON DELETE  
CASCADE);
```

```
CREATE TABLE User_Information(  
    user_id INT NOT NULL,  
    email VARCHAR(255) NOT NULL,  
    phone_number VARCHAR(10) NOT NULL,  
    password VARCHAR(255) NOT NULL,  
    PRIMARY KEY (user_id));
```

```
CREATE TABLE Forum(  
    post_id INT NOT NULL,  
    user_id INT NOT NULL,  
    post_timestamp VARCHAR(255) NOT NULL,  
    post_title VARCHAR(255) NOT NULL,  
    post_content VARCHAR(5000) NOT NULL,  
    PRIMARY KEY (post_id , user_id),  
    FOREIGN KEY(user_id) REFERENCES User_Information(user_id) ON DELETE  
CASCADE);
```

c. Inserting at least 1000 rows in the tables

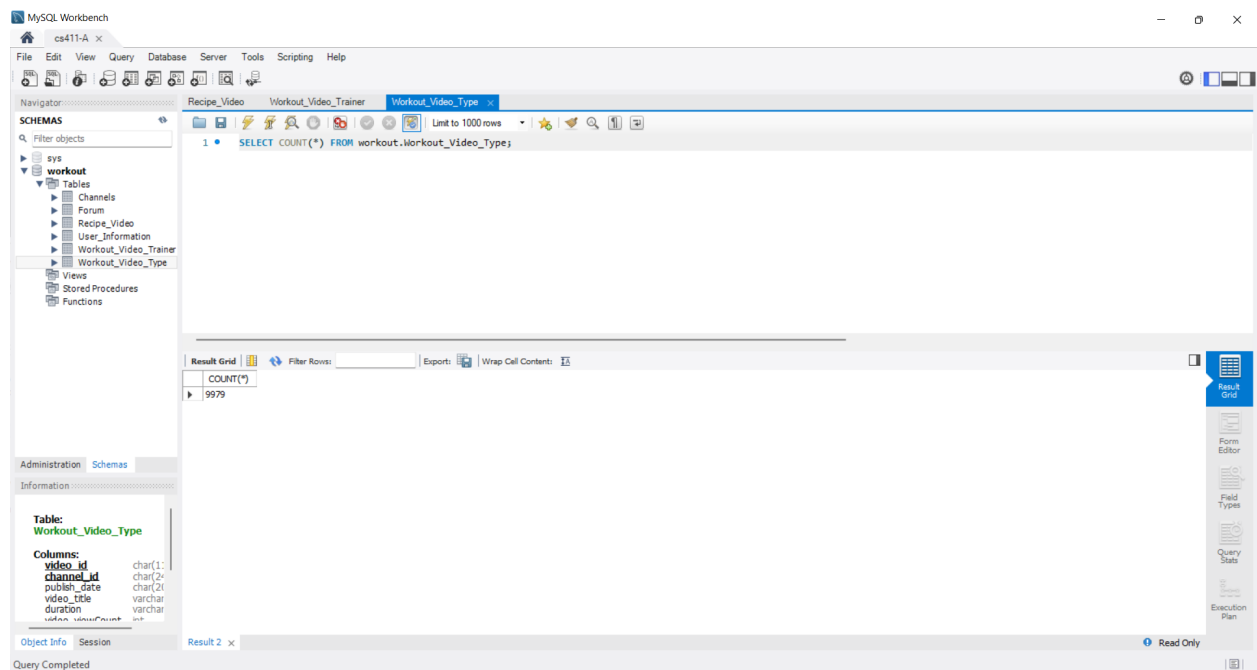
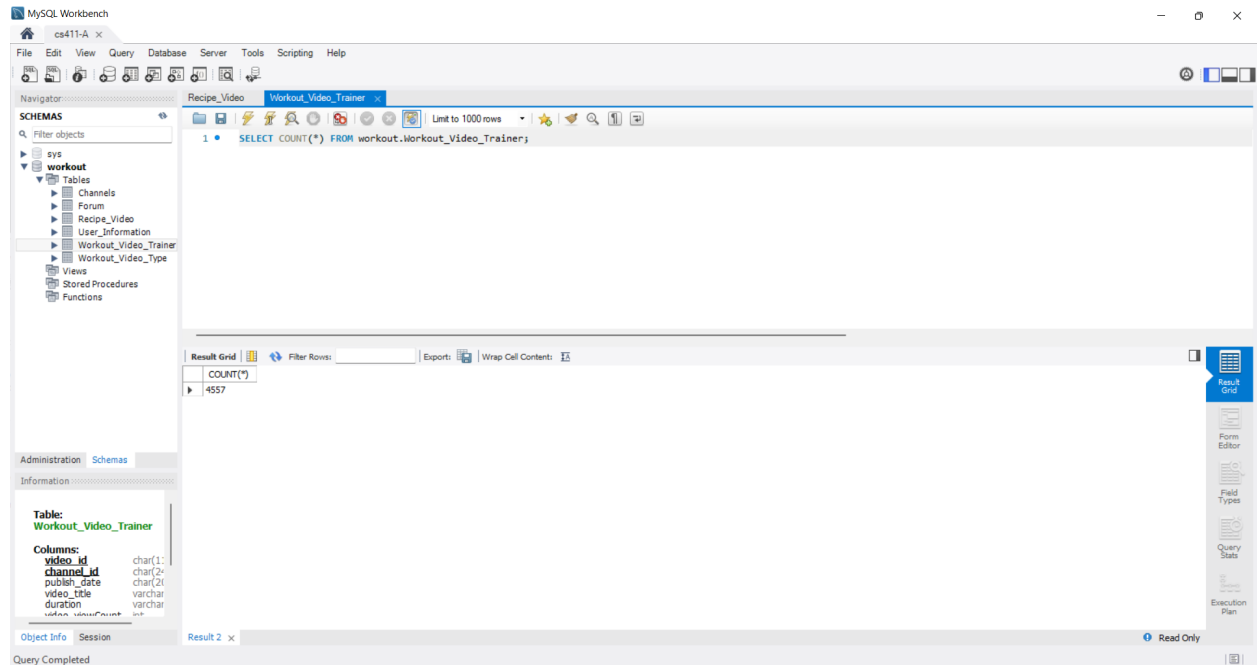
The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'workout' database selected. The 'Recipe_Video' table is highlighted. The main editor window shows a SQL query: `SELECT COUNT(*) FROM workout.Recipe_Video;`. The 'Result Grid' at the bottom shows a single row with the value 4301. The 'Object Info' panel on the left provides details about the 'Recipe_Video' table, including its columns and data types.

Table: Recipe_Video

Columns:

Column Name	Data Type
video_id	char(1)
channel_id	char(2)
publin_date	char(2)
video_title	varchar
duration	varchar
video_viewCount	int
likeCount	int

Query Completed



2. Advanced Queries

- a. If a user is interested in HIIT&crossfit and wants to know the number of million views video of each channel. For trainer videos, they want to filter the video publish after 2020 (inclusive).

(SELECT channel_title, COUNT(video_id) AS cnt
FROM workout.Workout_Video_Trainer NATURAL JOIN workout.Channels
WHERE video_viewCount > 1000000 AND publish_date LIKE "202%")

GROUP BY channel_id

UNION

```
SELECT channel_title, COUNT(video_id) AS cnt
FROM workout.Workout_Video_Type NATURAL JOIN workout.Channels
WHERE video_viewCount > 1000000 AND workout_type = "HIIT&crossfit"
GROUP BY channel_id
```

UNION

```
SELECT channel_title, COUNT(video_id) AS cnt
FROM workout.Recipe_Video NATURAL JOIN workout.Channels
WHERE video_viewCount > 1000000
GROUP BY channel_id)
ORDER BY cnt DESC
LIMIT 15;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains a query that combines two SELECT statements using a UNION. The first SELECT statement counts videos from the Workout_Video_Type table, and the second counts videos from the Recipe_Video table, both filtered by view count and workout type. The results grid shows the top 15 channels by video count.

channel_title	cnt
Chloe Ting	147
Pamela Reif	143
growingannanas	119
Skippy Recipes	94
CrossFit®	88
TheSeriousfitness	64
Caroline Girvan	51
Heather Robertson	45
Downshftology	44
Clean & Delicious	37
Mady Morrison	36
Dance With Deepti	18
Juice & Toya	13
Simeon Panda	12
Move With Nicole	11

- b. Suppose vegan people want to know the recipe videos which have at least 100000 views and the channel has millions of subscribers. We want to do subquery first and then join, to avoid the expensive cost of joins.

```
SELECT *
FROM (SELECT * FROM workout.Recipe_Video WHERE video_viewCount > 100000) AS rv
NATURAL JOIN (SELECT * FROM workout.Channels WHERE subscriberCount > 1000000) AS ch
WHERE video_title LIKE "%vegan%";
```

MySQL Workbench

cs411-A x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

sys

workout

Tables

Channels

Columns

channel_id

channel_title

subscriberCo

channel_view

videoCount

Indexes

Foreign Keys

Triggers

Forum

Recipe_Video

Columns

video_id

publish_date

video_title

duration

video_viewC

Administration Schemas

Information

Column: subscriberCount

Definition:

subscriberCount int

Query Completed

adv query2

Limit to 1000 rows

```

1 SELECT *
2 FROM (SELECT * FROM workout.Recipe_Video WHERE video_viewCount > 1000000) AS rv NATURAL JOIN (SELECT * FROM workout.Channels WHERE subscriberCount > 1000000) AS ch
3 WHERE video_title LIKE "%vegan%"
4 ORDER BY likeCount DESC
5 LIMIT 15;

```

Result Grid

channel_id	video_id	publish_date	video_title	duration	video_viewCount	likeCount	channel_title	subscriberCount	channel_viewCount	videoCount
UC_4kz6_R00xYwKt8PvA	qos2FvHnas	2018-11-29T11:36:13Z	4 Healthy Vegan Recipes For Weight Loss	PT10M28S	5301327	84335	TheSeriousfitness	1920000	273172618	270
UC_4kz6_R00xYwKt8PvA	zmhwTgHq4	2019-05-18T12:46:36Z	13 Healthy Vegan Recipes For Weight Loss	PT11M34S	3850142	30501	TheSeriousfitness	1920000	273172618	270
UCJN3h7QwQIF2_RHfKvveg	2w5d3aK3u	2018-10-02T09:37:15Z	Oats Breakfast Smoothie Recipe - Oats Recipes	PT3M38S	1956546	39373	Skinny Recipes	3120000	401440355	997
UCJN3h7QwQIF2_RHfKvveg	aQ00ndGqG0	2019-07-19T14:48:31Z	THE BEST FALAFEL RECIPE crispy fried and ba...	PT9M50S	1620622	36611	Downshifology	2370000	178539448	188
UCJN3h7QwQIF2_RHfKvveg	wpiFSPCLn8	2018-08-24T13:29:56Z	VEGAN PANCAKES Light + Fluffy Vegan Panca...	PT6M15S	1313035	31448	Clean & Delicious	1980000	207157750	779
UCJN3h7QwQIF2_RHfKvveg	QepGu2fORuJ	2019-02-07T09:24:01Z	Healthy Smoothie Recipes For Weight Loss - Ve...	PT9M65S	1147044	31413	Skinny Recipes	3120000	401440355	997
UCJN3h7QwQIF2_RHfKvveg	ABXQvXPGRUM	2020-03-01T15:01:34Z	FALAFEL FLATBREAD amazing vegan flatbrea...	PT8M14S	523340	27075	Downshifology	2370000	178539448	188
UCJN3h7QwQIF2_RHfKvveg	TQ8Rv-wdaJU	2018-09-09T16:16:37Z	HOW TO MAKE ALMOND MILK dairy-free, veg...	PT3M35S	851319	24984	Downshifology	2370000	178539448	188
UCJN3h7QwQIF2_RHfKvveg	4wzbKfne70	2017-11-16T19:01:24Z	VEGAN CARAMEL CHEESECAKE gluten-free ve...	PT9M21S	442322	18752	Downshifology	2370000	178539448	188
UCJN3h7QwQIF2_RHfKvveg	DvqJTEHk3v	2018-06-08T13:56:18Z	Chocolate Chip Brownie Bars Vegan + Paleo +...	PT9M44S	304646	17540	Clean & Delicious	1980000	207157750	779
UCJN3h7QwQIF2_RHfKvveg	Ma8L7HnQ3E	2020-06-19T11:53:44Z	How To Make High Protein Vegan Milk And Curd...	PT9M14S	401051	14042	Skinny Recipes	3120000	401440355	997
UCJN3h7QwQIF2_RHfKvveg	nXET17-a77E	2020-09-05T10:00:21Z	4 Vegan Curd Recipes - Dairy Free Curd - Home...	PT13M47S	497301	13540	Skinny Recipes	3120000	401440355	997
UCJN3h7QwQIF2_RHfKvveg	ghoCMmrbOTw	2016-10-05T22:57:43Z	HOW TO MAKE CASHEW MILK dairy-free, veg...	PT2M	358388	12586	Downshifology	2370000	178539448	188
UCJN3h7QwQIF2_RHfKvveg	naE27KpVvg	2020-11-08T08:00:05Z	Eggless Oats Cookies - No Oven, No Maida, No ...	PT6M27S	382672	10889	Skinny Recipes	3120000	401440355	997
UCJN3h7QwQIF2_RHfKvveg	pN5D-4nEtCI	2019-02-09T09:46:23Z	Dal Soup For Weight Loss - Healthy Lentil Soup ...	PT9M50S	516332	10721	Skinny Recipes	3120000	401440355	997

Object Info Session Result 13 x

Read Only

3. Indexing Analysis

a. Query1

-> Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.029..0.031 rows=15 loops=1)

-> Sort: cnt DESC, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.028..0.029 rows=15 loops=1)

-> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.005 rows=21 loops=1)

-> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=11.993..11.996 rows=21 loops=1)

-> Table scan on <temporary> (actual time=0.002..0.003 rows=10 loops=1)

-> Aggregate using temporary table (actual time=3.256..3.258 rows=10 loops=1)

-> Nested loop inner join (cost=562.96 rows=177) (actual time=0.109..2.933 rows=472 loops=1)

-> Filter: ((Workout_Video_Trainer.video_viewCount > 1000000) and (Workout_Video_Trainer.publish_date like '202%')) (cost=501.15 rows=177) (actual time=0.089..2.337 rows=472 loops=1)

-> Table scan on Workout_Video_Trainer (cost=501.15 rows=4769) (actual time=0.075..1.875 rows=4557 loops=1)

-> Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Video_Trainer.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=472)

-> Table scan on <temporary> (actual time=0.001..0.002 rows=3 loops=1)

-> Aggregate using temporary table (actual time=6.588..6.589 rows=3 loops=1)

-> Nested loop inner join (cost=1129.66 rows=325) (actual time=0.066..6.422 rows=220 loops=1)

-> Filter: ((Workout_Video_Type.workout_type = 'HIIT&crossfit') and (Workout_Video_Type.video_viewCount > 1000000)) (cost=1015.85 rows=325) (actual time=0.057..6.209 rows=220 loops=1)

-> Table scan on Workout_Video_Type (cost=1015.85 rows=9756) (actual time=0.050..3.640 rows=9979 loops=1)

-> Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Video_Type.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=220)

-> Table scan on <temporary> (actual time=0.001..0.002 rows=8 loops=1)

-> Aggregate using temporary table (actual time=2.089..2.090 rows=8 loops=1)

-> Nested loop inner join (cost=1039.50 rows=1562) (actual time=0.058..1.918 rows=248 loops=1)

-> Filter: (Recipe_Video.video_viewCount > 1000000) (cost=492.85 rows=1562) (actual time=0.052..1.639 rows=248 loops=1)

-> Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.048..1.396 rows=4301 loops=1)

-> Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=248)

CREATE INDEX video_viewCount_idx ON workout.Workout_Video_Trainer(video_viewCount);
CREATE INDEX video_viewCount_idx ON workout.Workout_Video_Type(video_viewCount);
CREATE INDEX video_viewCount_idx ON workout.Recipe_Video(video_viewCount);

-> Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.022..0.024 rows=15 loops=1)

-> Sort: cnt DESC, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.022..0.023 rows=15 loops=1)

-> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.003 rows=21 loops=1)

-> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=18.440..18.443 rows=21 loops=1)

-> Table scan on <temporary> (actual time=0.002..0.004 rows=10 loops=1)

-> Aggregate using temporary table (actual time=9.019..9.023 rows=10 loops=1)

-> Nested loop inner join (cost=374.75 rows=85) (actual time=0.399..8.649 rows=472 loops=1)

-> Filter: (Workout_Video_Trainer.publish_date like '202%') (cost=344.96 rows=85) (actual time=0.385..8.042 rows=472 loops=1)

-> Index range scan on Workout_Video_Trainer using video_viewCount_idx, with index condition: (Workout_Video_Trainer.video_viewCount > 1000000) (cost=344.96 rows=766) (actual time=0.381..7.896 rows=766 loops=1)

- > Single-row index lookup on Channels using PRIMARY
(channel_id=Workout_Video_Trainer.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=472)
 - > Table scan on <temporary> (actual time=0.001..0.002 rows=3 loops=1)
 - > Aggregate using temporary table (actual time=8.644..8.645 rows=3 loops=1)
 - > Nested loop inner join (cost=364.01 rows=75) (actual time=0.284..8.418 rows=220 loops=1)
 - > Filter: (Workout_Video_Type.workout_type = 'HIIT&crossfit') (cost=337.76 rows=75) (actual time=0.278..8.140 rows=220 loops=1)
 - > Index range scan on Workout_Video_Type using video_viewCount_idx, with index condition: (Workout_Video_Type.video_viewCount > 1000000) (cost=337.76 rows=750) (actual time=0.275..7.984 rows=750 loops=1)
 - > Single-row index lookup on Channels using PRIMARY
(channel_id=Workout_Video_Type.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=220)
 - > Table scan on <temporary> (actual time=0.001..0.002 rows=8 loops=1)
 - > Aggregate using temporary table (actual time=0.715..0.717 rows=8 loops=1)
 - > Nested loop inner join (cost=141.44 rows=248) (actual time=0.050..0.492 rows=248 loops=1)
 - > Filter: (Recipe_Video.video_viewCount > 1000000) (cost=54.64 rows=248) (actual time=0.043..0.164 rows=248 loops=1)
 - > Index range scan on Recipe_Video using video_viewCount_idx (cost=54.64 rows=248) (actual time=0.041..0.137 rows=248 loops=1)
 - > Single-row index lookup on Channels using PRIMARY
(channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=248)

CREATE INDEX publish_date_idx ON workout.Workout_Video_Trainer(publish_date);

- > Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.037..0.039 rows=15 loops=1)
 - > Sort: cnt DESC, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.028..0.029 rows=15 loops=1)
 - > Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.002..0.006 rows=21 loops=1)
 - > Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=12.368..12.370 rows=21 loops=1)
 - > Table scan on <temporary> (actual time=0.001..0.002 rows=10 loops=1)
 - > Aggregate using temporary table (actual time=3.120..3.121 rows=10 loops=1)
 - > Nested loop inner join (cost=779.26 rows=795) (actual time=0.090..2.795 rows=472 loops=1)
 - > Filter: ((Workout_Video_Trainer.video_viewCount > 1000000) and (Workout_Video_Trainer.publish_date like '202%')) (cost=501.15 rows=795) (actual time=0.075..2.214 rows=472 loops=1)

- > Table scan on Workout_Video_Trainer (cost=501.15 rows=4769) (actual time=0.063..1.776 rows=4557 loops=1)
- > Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Video_Trainer.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=472)
- > Table scan on <temporary> (actual time=0.001..0.002 rows=3 loops=1)
- > Aggregate using temporary table (actual time=7.077..7.077 rows=3 loops=1)
- > Nested loop inner join (cost=1129.66 rows=325) (actual time=0.049..6.885 rows=220 loops=1)
- > Filter: ((Workout_Video_Type.workout_type = 'HIIT&crossfit') and (Workout_Video_Type.video_viewCount > 1000000)) (cost=1015.85 rows=325) (actual time=0.044..6.654 rows=220 loops=1)
- > Table scan on Workout_Video_Type (cost=1015.85 rows=9756) (actual time=0.037..3.813 rows=9979 loops=1)
- > Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Video_Type.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=220)
- > Table scan on <temporary> (actual time=0.001..0.002 rows=8 loops=1)
- > Aggregate using temporary table (actual time=2.104..2.105 rows=8 loops=1)
- > Nested loop inner join (cost=1039.50 rows=1562) (actual time=0.062..1.926 rows=248 loops=1)
- > Filter: (Recipe_Video.video_viewCount > 1000000) (cost=492.85 rows=1562) (actual time=0.053..1.637 rows=248 loops=1)
- > Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.050..1.381 rows=4301 loops=1)
- > Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=248)

CREATE INDEX workout_type_idx ON workout.Workout_Video_Type(workout_type);

- > Limit: 15 row(s) (cost=2.50 rows=0) (actual time=0.023..0.025 rows=15 loops=1)
- > Sort: cnt DESC, limit input to 15 row(s) per chunk (cost=2.50 rows=0) (actual time=0.023..0.023 rows=15 loops=1)
- > Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.004 rows=21 loops=1)
- > Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=24.575..24.578 rows=21 loops=1)
- > Table scan on <temporary> (actual time=0.001..0.002 rows=10 loops=1)
- > Aggregate using temporary table (actual time=3.177..3.179 rows=10 loops=1)
- > Nested loop inner join (cost=562.96 rows=177) (actual time=0.086..2.832 rows=472 loops=1)

-> Filter: ((Workout_Video_Trainer.video_viewCount > 1000000) and (Workout_Video_Trainer.publish_date like '202%')) (cost=501.15 rows=177) (actual time=0.071..2.257 rows=472 loops=1)

-> Table scan on Workout_Video_Trainer (cost=501.15 rows=4769) (actual time=0.058..1.800 rows=4557 loops=1)

-> Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Video_Trainer.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=472)

-> Table scan on <temporary> (actual time=0.002..0.003 rows=3 loops=1)

-> Aggregate using temporary table (actual time=18.684..18.684 rows=3 loops=1)

-> Nested loop inner join (cost=852.38 rows=1626) (actual time=0.156..18.463 rows=220 loops=1)

-> Filter: (Workout_Video_Type.video_viewCount > 1000000) (cost=283.33 rows=1626) (actual time=0.150..18.183 rows=220 loops=1)

-> Index lookup on Workout_Video_Type using workout_type_idx (workout_type='HIIT&crossfit') (cost=283.33 rows=4878) (actual time=0.146..17.748 rows=7416 loops=1)

-> Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Video_Type.channel_id) (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=220)

-> Table scan on <temporary> (actual time=0.001..0.002 rows=8 loops=1)

-> Aggregate using temporary table (actual time=2.653..2.655 rows=8 loops=1)

-> Nested loop inner join (cost=1039.50 rows=1562) (actual time=0.055..2.442 rows=248 loops=1)

-> Filter: (Recipe_Video.video_viewCount > 1000000) (cost=492.85 rows=1562) (actual time=0.048..1.928 rows=248 loops=1)

-> Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.044..1.653 rows=4301 loops=1)

-> Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=248)

Filter: ((Workout_Video_Trainer.video_viewCount > 1000000) and (Workout_Video_Trainer.publish_date like '202%')) (cost=501.15 rows=177) (actual time=0.089..2.337 rows=472 loops=1)

Filter: ((Workout_Video_Type.workout_type = 'HIIT&crossfit') and (Workout_Video_Type.video_viewCount > 1000000)) (cost=1015.85 rows=325) (actual time=0.057..6.209 rows=220 loops=1)

Filter: (Recipe_Video.video_viewCount > 1000000) (cost=492.85 rows=1562) (actual time=0.052..1.639 rows=248 loops=1)

Before adding the index on video_viewCount, the cost of each subquery is 501.15, 1015.85, and 492.85 separately. After adding the index on video_viewCount, the cost reduces to

344.96, 337.76, and 54.64, which improves the performance by over 60%. Adding the index on publish_date doesn't affect the analyzed result because indexing cannot be used on forms of LIKE "abc%". On the other hand, adding the index on workout_type is very successful, reducing cost from 1015.85 to 283.33, decreasing over 70% of the cost. This is because we only have four types of workout_type, so indexing on it would be very useful.

b. Query2

-> Limit: 15 row(s) (cost=1004.83 rows=15) (actual time=5.650..5.684 rows=15 loops=1)
-> Nested loop inner join (cost=1004.83 rows=1562) (actual time=5.649..5.682 rows=15 loops=1)
-> Sort: Recipe_Video.likeCount DESC (cost=492.85 rows=4686) (actual time=5.614..5.617 rows=19 loops=1)
-> Filter: ((Recipe_Video.video_title like '%vegan%') and (Recipe_Video.video_viewCount > 100000)) (cost=492.85 rows=4686) (actual time=0.197..5.556 rows=98 loops=1)
-> Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.086..2.080 rows=4301 loops=1)
-> Filter: (Channels.subscriberCount > 1000000) (cost=0.25 rows=0) (actual time=0.003..0.003 rows=1 loops=19)
-> Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=19)

CREATE INDEX video_viewCount_idx ON workout.Recipe_Video(video_viewCount);

-> Limit: 15 row(s) (cost=1001.50 rows=15) (actual time=5.610..5.645 rows=15 loops=1)
-> Nested loop inner join (cost=1001.50 rows=1562) (actual time=5.609..5.643 rows=15 loops=1)
-> Sort: Recipe_Video.likeCount DESC (cost=492.85 rows=4686) (actual time=5.576..5.579 rows=19 loops=1)
-> Filter: ((Recipe_Video.video_title like '%vegan%') and (Recipe_Video.video_viewCount > 100000)) (cost=492.85 rows=4686) (actual time=0.225..5.518 rows=98 loops=1)
-> Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.072..2.057 rows=4301 loops=1)
-> Filter: (Channels.subscriberCount > 1000000) (cost=0.25 rows=0) (actual time=0.003..0.003 rows=1 loops=19)
-> Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=19)

CREATE INDEX subscriberCount_idx ON workout.Channels(subscriberCount);

-> Limit: 15 row(s) (cost=1004.83 rows=15) (actual time=5.657..5.691 rows=15 loops=1)
 -> Nested loop inner join (cost=1004.83 rows=2570) (actual time=5.656..5.689 rows=15 loops=1)
 -> Sort: Recipe_Video.likeCount DESC (cost=492.85 rows=4686) (actual time=5.626..5.629 rows=19 loops=1)
 -> Filter: ((Recipe_Video.video_title like '%vegan%') and (Recipe_Video.video_viewCount > 100000)) (cost=492.85 rows=4686) (actual time=0.185..5.568 rows=98 loops=1)
 -> Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.066..2.076 rows=4301 loops=1)
 -> Filter: (Channels.subscriberCount > 1000000) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=19)
 -> Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=19)

CREATE INDEX video_title_idx ON workout.Recipe_Video(video_title);

-> Limit: 15 row(s) (cost=1004.83 rows=15) (actual time=6.060..6.096 rows=15 loops=1)
 -> Nested loop inner join (cost=1004.83 rows=1562) (actual time=6.059..6.094 rows=15 loops=1)
 -> Sort: Recipe_Video.likeCount DESC (cost=492.85 rows=4686) (actual time=6.019..6.022 rows=19 loops=1)
 -> Filter: ((Recipe_Video.video_title like '%vegan%') and (Recipe_Video.video_viewCount > 100000)) (cost=492.85 rows=4686) (actual time=0.185..5.953 rows=98 loops=1)
 -> Table scan on Recipe_Video (cost=492.85 rows=4686) (actual time=0.085..2.207 rows=4301 loops=1)
 -> Filter: (Channels.subscriberCount > 1000000) (cost=0.25 rows=0) (actual time=0.003..0.004 rows=1 loops=19)
 -> Single-row index lookup on Channels using PRIMARY (channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.003..0.003 rows=1 loops=19)

Because we use LIKE “%abc%” in the where clause, any indexing on Recipe_Video would be useless. Moreover, every subscriberCount only has three significant digits, which causes them usually end with zeros, so indexing on subscriberCount is useless as well. In conclusion, three kinds of indexing do not affect the analyzed result, which means they do not work at all.