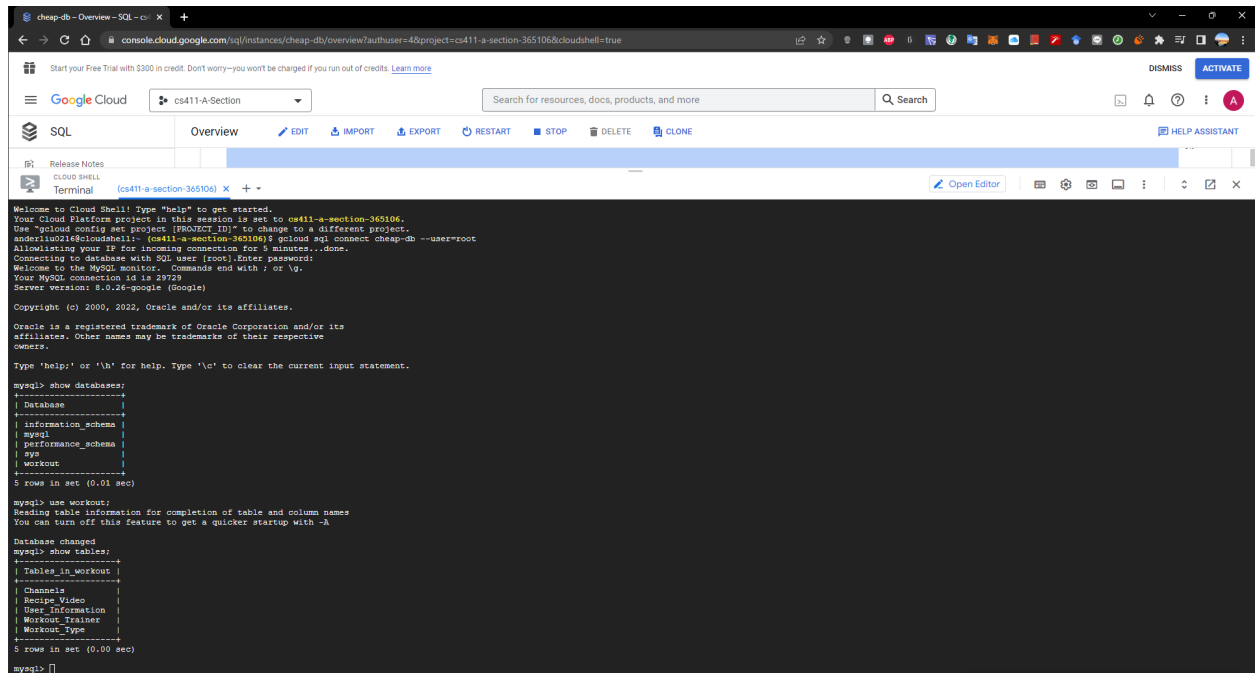


Project Stage 3

In project stage 3, we created a database with four tables, including a table for channel information, a table for workout videos based on workout type, a table for workout videos based on trainers, and a table for recipe videos.

1. **Submission location:** located within the doc folder
2. **Database implementation:**
 - a. **MySQL@GCP**



```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| workout |
+-----+
5 rows in set (0.01 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 |
| Recipe_Video |
| User_Information |
| Workout_Trainer |
| Workout_Type |
+-----+
5 rows in set (0.00 sec)

mysql>
```

- b. **DDL (as shown below):**

Data Definition Language (DDL)

create database workout;

use workout;

```
CREATE TABLE Channels (
channel_id INT NOT NULL,
channel_name VARCHAR(255) NOT NULL,
subscriber_num INT NOT NULL,
PRIMARY KEY (channel_id)
);
```

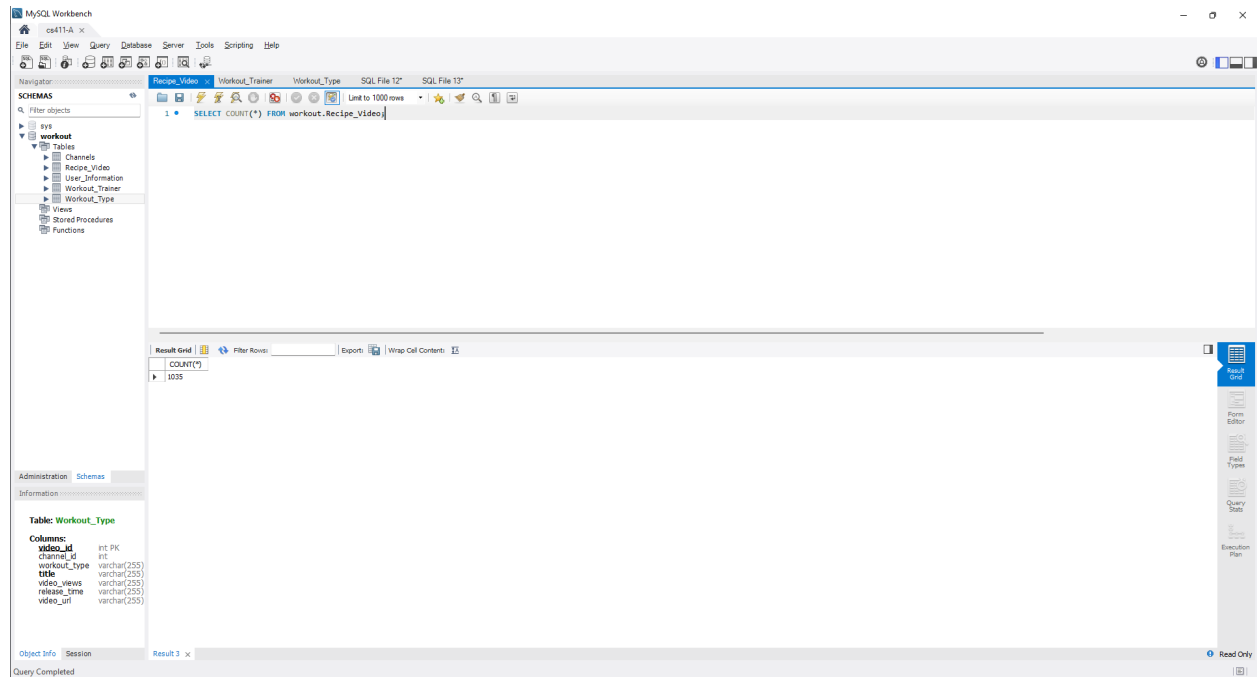
```
CREATE TABLE Workout_Type (
video_id INT NOT NULL,
channel_id INT NOT NULL,
```

```
workout_type VARCHAR(255),  
title VARCHAR(255),  
video_views VARCHAR(255),  
release_time VARCHAR(255),  
video_url VARCHAR(255),  
PRIMARY KEY (video_id)  
);
```

```
CREATE TABLE Workout_Trainer (  
video_id INT NOT NULL,  
channel_id INT NOT NULL,  
title VARCHAR(255),  
video_views VARCHAR(255),  
release_time VARCHAR(255),  
video_url VARCHAR(255),  
PRIMARY KEY (video_id)  
);
```

```
CREATE TABLE Recipe_Video (  
video_id INT NOT NULL,  
channel_id INT NOT NULL,  
title VARCHAR(255),  
video_views VARCHAR(255),  
release_time VARCHAR(255),  
video_url VARCHAR(255),  
PRIMARY KEY (video_id)  
);
```

c. **Insert at least 1000 rows each in three of the tables** (screenshot found below):



MySQL Workbench

cs411-A x

File Edit View Query Database Server Tools Scripting Help

Navigation

Schemas

Filter objects

sys

workout

Tables

Channels

Recipe_Video

User_Information

Workout_Trainer

Workout_Type

Views

Stored Procedures

Functions

Administration Schemas

Information

Table: Workout_Trainer

Columns:

video_id int PK

channel_id int

title varchar(255)

video_views varchar(255)

release_time varchar(255)

video_url varchar(255)

Object Info Session

Query Completed

Recipe_Video Workout_Trainer

1 * SELECT COUNT(*) FROM workout.Workout_Trainer;

Result Grid

Filter Rows:

Exports

Wrap Cell Contents

COUNT(*)

1000

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	22:38:19	SELECT * FROM workout.Recipe_Video LIMIT 0, 1000	1000 row(s) returned	0.016 sec / 0.031 sec
2	22:38:28	SELECT COUNT(*) FROM workout.Recipe_Video LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
3	23:07:05	SELECT * FROM workout.Workout_Trainer LIMIT 0, 1000	1000 row(s) returned	0.015 sec / 0.016 sec
4	23:07:12	SELECT COUNT(*) FROM workout.Workout_Trainer LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

MySQL Workbench

cs411-A x

File Edit View Query Database Server Tools Scripting Help

Navigation

Schemas

Filter objects

sys

workout

Tables

Channels

Recipe_Video

User_Information

Workout_Trainer

Workout_Type

Views

Stored Procedures

Functions

Administration Schemas

Information

Table: Workout_Type

Columns:

video_id int PK

channel_id int

workout_type varchar(255)

title varchar(255)

video_views varchar(255)

release_time varchar(255)

video_url varchar(255)

Object Info Session

Query Completed

Recipe_Video Workout_Trainer Workout_Type

1 * SELECT COUNT(*) FROM workout.Workout_Type;

Result Grid

Filter Rows:

Exports

Wrap Cell Contents

COUNT(*)

1000

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	22:38:19	SELECT * FROM workout.Recipe_Video LIMIT 0, 1000	1000 row(s) returned	0.016 sec / 0.031 sec
2	22:38:28	SELECT COUNT(*) FROM workout.Recipe_Video LIMIT 0, 1000	1 row(s) returned	0.015 sec / 0.000 sec
3	23:07:05	SELECT * FROM workout.Workout_Trainer LIMIT 0, 1000	1000 row(s) returned	0.015 sec / 0.016 sec
4	23:07:12	SELECT COUNT(*) FROM workout.Workout_Trainer LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec
5	23:07:25	SELECT * FROM workout.Workout_Type LIMIT 0, 1000	1000 row(s) returned	0.016 sec / 0.000 sec
6	23:07:31	SELECT COUNT(*) FROM workout.Workout_Type LIMIT 0, 1000	1 row(s) returned	0.016 sec / 0.000 sec

3. Advance queries

- a. **Query 1:** For the first query, we seek to find channels with video views over 1 million in each of the three video tables, and union the three tables.

SQL Query1:

```
SELECT channel_name, COUNT(video_id) AS cnt
FROM workout.Workout_Type NATURAL JOIN workout.Channels
WHERE video_views LIKE '%M views'
GROUP BY channel_id
```

UNION

```
SELECT channel_name, COUNT(video_id) AS cnt
FROM workout.Workout_Trainer NATURAL JOIN workout.Channels
WHERE video_views LIKE '%M views'
GROUP BY channel_id
```

UNION

```
SELECT channel_name, COUNT(video_id) AS cnt
FROM workout.Recipe_Video NATURAL JOIN workout.Channels
WHERE video_views LIKE '%M views'
GROUP BY channel_id
```

```
ORDER BY cnt DESC
LIMIT 15;
```

Screenshot of Query 1 (first 15 rows):

The screenshot displays the MySQL Workbench interface. The SQL editor at the top contains the following query:

```
1 SELECT channel_name, COUNT(video_id) AS cnt
2 FROM workout.Workout_Type NATURAL JOIN workout.Channels
3 WHERE video_views LIKE '%M views'
4 GROUP BY channel_id
5
6 UNION
7
8 SELECT channel_name, COUNT(video_id) AS cnt
9 FROM workout.Workout_Trainer NATURAL JOIN workout.Channels
10 WHERE video_views LIKE '%M views'
11 GROUP BY channel_id
12
```

The 'Result Grid' at the bottom shows the first 15 rows of the query results:

channel_name	cnt
Yoga With Adriene	79
Mady Morrison	71
Chloe Ting	62
growingannanas	38
Dance With Deepti	38
Jesse James Frey	9
Simon Panda	8
AEROBIC DANCE	5
Soho Beautiful Yoga	2
YOGABODY	2
Aerobic Workout	1
Lucy Wyndham-Read	1
Caroline Girvan	1

The 'Table: Workout_Type' section shows the table structure:

Column	DataType	PK
video_id	int	PK
channel_id	int	PK
workout_type	varchar(255)	
title	varchar(255)	
video_views	varchar(255)	
release_time	varchar(255)	
video_url	varchar(255)	

The 'Action Output' section shows the execution progress of the query, including the number of rows returned and the duration of each step.

- b. **Query 2:** For the second query, we aim to find all the recipe videos that posted vegan recipes from channels that have subscribers at a Million-subscriber level. To achieve that, we joined Recipe_Video and Channels, using subquery to find channels with millions of subscribers and “LIKE” to find vegan recipes.

SQL Query2:

SELECT title

FROM (SELECT * FROM workout.Recipe_Video WHERE video_views > 20000)

vv NATURAL JOIN (SELECT * FROM workout.Channels WHERE

subscriber_num > 1000000) cc

WHERE title LIKE "%vegan%"

LIMIT 15

Screenshot of Query 2 (first 15 rows):

The screenshot shows the MySQL Workbench interface. The SQL editor at the top contains the following query:

```
1 SELECT title
2 FROM (SELECT * FROM workout.Recipe_Video WHERE video_views > 20000) vv NATURAL JOIN (SELECT * FROM workout.Channels WHERE
3 subscriber_num > 1000000) cc
4 WHERE title LIKE "%vegan%"
5 LIMIT 15;
```

The Result Grid below the editor displays the first 15 rows of the query results. The columns are 'title' and 'video_views'. The results are as follows:

title	video_views
5 Healthy Vegan Recipes For Weight Loss	1000
13 Healthy Vegan Recipes For Weight Loss	1000
4 Healthy Vegan Recipes For Weight Loss	1000
3 Easy Vegan Recipes For Weight Loss	1000
Chocolate Oat Milk Recipe - "Schokolade" alt: "Dessert"	1000
Avocado Pudding - No Sugar - No Dairy Milk - "Dessert"	1000
Chia Pudding - 2 Easy & Healthy Chia Pudding Recipes	1000
Dairy Free Hot Chocolate Recipe - Vegan Hot Chocolate	1000
2 Healthy Rag Recipes For Weight Loss - Rag Recipes	1000
NO OIL Salad Recipe For Weight loss - Veg Salad	1000
High Protein Salad Recipe - Weight Loss Salad Recipe	1000
Mango Milk Recipe - No Dairy Milk - No Sugar - Vegan	1000
Weight Loss Summer Drink - No Dairy Milk - No Sugar	1000
MINI PECAN PIE / No bake, vegan recipe	1000
BEST BLUEBERRY CRISP vegan recipe	1000

The bottom of the screenshot shows the 'Table: Workout_Type' structure and the 'Output' tab, which displays the execution plan for the query.

4. Indexing Analysis

a. Indexing Analysis for Query1

For the first query, we seek to find channels with video views over 1 million in each of the three video tables, and union the three tables.

-> Sort: cnt DESC (cost=2.50 rows=0) (actual time=0.020..0.021 rows=18 loops=1)
-> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.003 rows=18 loops=1)
-> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual time=2.838..2.840 rows=18 loops=1)
-> Table scan on <temporary> (actual time=0.001..0.001 rows=7 loops=1)
-> Aggregate using temporary table (actual time=0.824..0.825 rows=7 loops=1)
-> Nested loop inner join (cost=142.14 rows=111) (actual time=0.150..0.736 rows=136 loops=1)
-> Filter: (Workout_Type.video_views like '%M views') (cost=103.25 rows=111) (actual time=0.134..0.656 rows=136 loops=1)
-> Table scan on Workout_Type (cost=103.25 rows=1000) (actual time=0.056..0.393 rows=1000 loops=1)
-> Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Type.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=136)
-> Table scan on <temporary> (actual time=0.001..0.002 rows=6 loops=1)
-> Aggregate using temporary table (actual time=1.281..1.282 rows=6 loops=1)
-> Nested loop inner join (cost=145.00 rows=112) (actual time=0.246..1.128 rows=161 loops=1)
-> Filter: (Workout_Trainer.video_views like '%M views') (cost=105.80 rows=112) (actual time=0.237..1.035 rows=161 loops=1)
-> Table scan on Workout_Trainer (cost=105.80 rows=1008) (actual time=0.044..0.622 rows=1008 loops=1)
-> Single-row index lookup on Channels using PRIMARY (channel_id=Workout_Trainer.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=161)
-> Table scan on <temporary> (actual time=0.000..0.001 rows=5 loops=1)
-> Aggregate using temporary table (actual time=0.687..0.688 rows=5 loops=1)
-> Nested loop inner join (cost=147.75 rows=115) (actual time=0.133..0.636 rows=80 loops=1)
-> Filter: (Recipe_Video.video_views like '%M views') (cost=107.50 rows=115) (actual time=0.126..0.596 rows=80 loops=1)
-> Table scan on Recipe_Video (cost=107.50 rows=1035) (actual time=0.057..0.402 rows=1035 loops=1)

-> Single-row index lookup on Channels using PRIMARY
(channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1
loops=80)

CREATE INDEX video_views_idx ON workout.Recipe_Video(video_views);

-> Sort: cnt DESC (cost=2.50 rows=0) (actual time=0.023..0.024 rows=18 loops=1)
-> Table scan on <union temporary> (cost=2.50 rows=0) (actual time=0.001..0.004 rows=18
loops=1)

-> Union materialize with deduplication (cost=2.50..2.50 rows=0) (actual
time=2.329..2.331 rows=18 loops=1)

-> Table scan on <temporary> (actual time=0.001..0.002 rows=7 loops=1)

-> Aggregate using temporary table (actual time=0.828..0.829 rows=7 loops=1)

-> Nested loop inner join (cost=142.14 rows=111) (actual time=0.219..0.746
rows=136 loops=1)

-> Filter: (Workout_Type.video_views like '%M views') (cost=103.25 rows=111)
(actual time=0.200..0.675 rows=136 loops=1)

-> Table scan on Workout_Type (cost=103.25 rows=1000) (actual
time=0.060..0.456 rows=1000 loops=1)

-> Single-row index lookup on Channels using PRIMARY
(channel_id=Workout_Type.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1
loops=136)

-> Table scan on <temporary> (actual time=0.000..0.001 rows=6 loops=1)

-> Aggregate using temporary table (actual time=0.735..0.736 rows=6 loops=1)

-> Nested loop inner join (cost=145.00 rows=112) (actual time=0.203..0.628
rows=161 loops=1)

-> Filter: (Workout_Trainer.video_views like '%M views') (cost=105.80 rows=112)
(actual time=0.195..0.566 rows=161 loops=1)

-> Table scan on Workout_Trainer (cost=105.80 rows=1008) (actual
time=0.062..0.375 rows=1008 loops=1)

-> Single-row index lookup on Channels using PRIMARY
(channel_id=Workout_Trainer.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000
rows=1 loops=161)

-> Table scan on <temporary> (actual time=0.001..0.001 rows=5 loops=1)

-> Aggregate using temporary table (actual time=0.705..0.706 rows=5 loops=1)

-> Nested loop inner join (cost=147.75 rows=115) (actual time=0.119..0.630
rows=80 loops=1)

-> Filter: (Recipe_Video.video_views like '%M views') (cost=107.50 rows=115)
(actual time=0.112..0.585 rows=80 loops=1)

-> Table scan on Recipe_Video (cost=107.50 rows=1035) (actual
time=0.047..0.365 rows=1035 loops=1)

-> Single-row index lookup on Channels using PRIMARY
(channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=80)

After using indexing, the cost retained unchanged. The result indicates no improvements in indexing. Since we used forms of LIKE “%abc%” and LIKE “%abc”, indexing cannot be used. Therefore, no improvements were identified. While “abc%” can be used for indexing.

b. Indexing analysis for Query2

For the second query, we aim to find all the recipe videos that posted vegan recipes from channels that have subscribers at a Million-subscriber level. To achieve that, we joined Recipe_Video and Channels, using subquery to find channels with millions of subscribers and “LIKE” to find vegan recipes.

-> Limit: 15 row(s) (cost=121.91 rows=15) (actual time=0.420..1.376 rows=15 loops=1)
-> Nested loop inner join (cost=121.91 rows=19) (actual time=0.419..1.373 rows=15 loops=1)
-> Filter: ((Recipe_Video.title like '%vegan%') and (Recipe_Video.video_views > 20000)) (cost=108.50 rows=38) (actual time=0.201..1.312 rows=75 loops=1)
-> Table scan on Recipe_Video (cost=108.50 rows=1035) (actual time=0.182..0.535 rows=1031 loops=1)
-> Filter: (Channels.subscriber_num > 1000000) (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=75)
-> Single-row index lookup on Channels using PRIMARY
(channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1 loops=75)

CREATE INDEX video_views_idx ON workout.Recipe_Video(video_views);

-> Limit: 15 row(s) (cost=121.91 rows=15) (actual time=0.279..1.257 rows=15 loops=1)
-> Nested loop inner join (cost=121.91 rows=19) (actual time=0.278..1.255 rows=15 loops=1)
-> Filter: ((Recipe_Video.title like '%vegan%') and (Recipe_Video.video_views > 20000)) (cost=108.50 rows=38) (actual time=0.088..1.202 rows=75 loops=1)
-> Table scan on Recipe_Video (cost=108.50 rows=1035) (actual time=0.072..0.434 rows=1031 loops=1)
-> Filter: (Channels.subscriber_num > 1000000) (cost=0.25 rows=0) (actual time=0.001..0.001 rows=0 loops=75)

-> Single-row index lookup on Channels using PRIMARY
(channel_id=Recipe_Video.channel_id) (cost=0.25 rows=1) (actual time=0.000..0.000 rows=1
loops=75)

CREATE INDEX subscriber_num_idx ON workout.Channels(subscriber_num);

-> Limit: 15 row(s) (cost=121.91 rows=15) (actual time=0.298..1.391 rows=15 loops=1)
-> Nested loop inner join (cost=121.91 rows=19) (actual time=0.297..1.388 rows=15
loops=1)
-> Filter: ((Recipe_Video.title like '%vegan%') and (Recipe_Video.video_views > 20000))
(cost=108.50 rows=38) (actual time=0.104..1.280 rows=75 loops=1)
-> Table scan on Recipe_Video (cost=108.50 rows=1035) (actual time=0.089..0.486
rows=1031 loops=1)
-> Filter: (Channels.subscriber_num > 1000000) (cost=0.25 rows=0) (actual
time=0.001..0.001 rows=0 loops=75)
-> 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=75)

CREATE INDEX title_idx ON workout.Recipe_Video(title);

-> Limit: 15 row(s) (cost=121.91 rows=13) (actual time=0.305..1.354 rows=15 loops=1)
-> Nested loop inner join (cost=121.91 rows=13) (actual time=0.304..1.352 rows=15
loops=1)
-> Filter: ((Recipe_Video.title like '%vegan%') and (Recipe_Video.video_views > 20000))
(cost=108.50 rows=38) (actual time=0.107..1.283 rows=75 loops=1)
-> Table scan on Recipe_Video (cost=108.50 rows=1035) (actual time=0.090..0.445
rows=1031 loops=1)
-> Filter: (Channels.subscriber_num > 1000000) (cost=0.25 rows=0) (actual
time=0.001..0.001 rows=0 loops=75)
-> 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=75)

Whether using indexing on workout.Recipe_Video(video_views) or
workout.Channels(subscriber_num), the cost retained unchanged. The result indicates no
improvements in indexing.

After using indexing on workout.Recipe_Video(title), the cost retained unchanged. The result
indicates no improvements in indexing. Since we used forms of LIKE “%abc%”, indexing cannot
be used. Therefore, no improvements were identified. While “abc%” can be used for indexing.

After using indexing on the second query twice, the cost when indexing "Recipe_Video.title like '%vegan%'" remains unchanged: the cost is 147.75 for the first time and 147.75 for the second time. The cost when indexing workout.Recipe_Video(video_views) also remains unchanged: 108.50 for the first time and 108.50 for the second time. Indexing on workout.Channels(subscriber_num) also remained unchanged: 121.91 for both first and second time. The result indicates no improvements in indexing.