CNIT 372 Final Project: Usage of Youtube as A Content Creator

CNIT - 372

Matthew Poole & Dawson Spratley

Submitted December 15th, 2023

Background

Our current undertaking involves a comprehensive examination aimed at elucidating the underlying motivations driving viewer engagement on the YouTube platform. This endeavor encompasses a meticulous statistical analysis of viewership trends, coupled with the integration of ancillary data points to ensure a holistic and nuanced understanding of user behavior.

Beyond a mere scrutiny of quantitative metrics, our approach entails discerning patterns and correlations within the data landscape. This analytical rigor is pivotal in unveiling the intricate motivations steering viewers toward specific content. Moreover, this inquiry extends beyond the immediate horizon, holding the potential to inform future content development strategies.

Of particular interest is the prospect of leveraging insights derived from our analytical endeavors to optimize content creation. By strategically incorporating keywords that emerge as influential in viewer engagement, we aspire to refine our creative output. This methodical integration of statistical analysis and strategic keyword utilization not only addresses current viewer motivations but also positions us strategically for a future characterized by sophisticated and tailored content creation.

Database Description

Our database features 3 tables. First, we created a table named Video that stores the videos that we watched. This table includes the video name, which is used as a primary key, the length of the video, the total number of likes and dislikes, the total number of views, the name of the channel who posted the video, the date that the video was posted, the number of times our individual accounts have viewed each video, and the date of each watch. Next, we created a table named Channel that stores information about the channel that the video came from. This table includes the channel name, which is used as a primary key in the Channel table and as a foreign key in the Video table, the number of videos that the channel has posted, the year that the channel was created, and the total number of views that the channel has amassed. Finally, we created our third and final table, the Viewer table. This table includes information about our own YouTube accounts. The information used was our user name, which was used as a primary key, the date that the account was created, the total number of videos that the viewer has viewed, the total number of videos that the viewer has either liked or disliked, and the video name of all videos that the viewer viewed, which is a foreign key in the table. There are 2 relationships in our database. The Video table acts as a parent for the Viewer table and the Channel table acts as a parent for the Video table.

Table Outline

Table	Attributes	Relationship
Video	**Video Name(PK)**, Length, Likes/Dislikes, Views, **Channel Name(FK)**, Date of Video Posted, View Count, Date Watched	Video Name (**PK**) Channel Name (**FK**)
Channel	**Channel Name(PK)**, Number of Videos, Year Created, Number of Views	Channel Name (**PK**)
Viewer	**User Name(PK)**, Date of Account Creation, Number of Videos Viewed, Number of Liked/Disliked Videos, **Video Name(FK)**	User Name (**PK**) Video Name (**FK**)

(Table Outline)

Solutions

Question 1

What is the channel name, the number of videos posted, and the average number of likes and dislikes for each channel watched?

This question can inform the user of what the distribution of likes/dislikes are across each channel with differing numbers of videos that we watch.

```
FUNCTION get_channel_info RETURN SYS_REFCURSOR IS
  channel_cursor SYS_REFCURSOR;
BEGIN
  OPEN channel_cursor FOR
    SELECT
      c.channel_name,
      COUNT(v.video_name) AS num_videos,
      AVG(v.likes_dislikes) AS avg_likes_dislikes
    FROM
      channel c
      JOIN video v ON c.channel_name = v.channel_name
    GROUP BY
      c.channel_name;

  RETURN channel_cursor;
END get_channel_info;
```

Question 2

How many videos were watched from each channel? Use the results from Question 1.

This question takes the information from the last question and provides a cursor containing information about each channel and the number of videos watched on each channel. This is to see if there is a correlation between the popularity of the channel and how many videos we watched.

```
FUNCTION count_videos_watched RETURN SYS_REFCURSOR IS
  watched_cursor SYS_REFCURSOR;
BEGIN
  OPEN watched_cursor FOR
    SELECT
      c.channel_name,
      COUNT(v.video_name) AS num_videos_watched
    FROM
      channel c
      JOIN video v ON c.channel_name = v.channel_name
    WHERE
      v.date_watched IS NOT NULL
    GROUP BY
      c.channel_name;

  RETURN watched_cursor;
END count_videos_watched;
```

Question 3

What is the date of each account's creation, the number of videos viewed from each channel, and the date that each video was watched?

The purpose of this function is to retrieve information about viewers, including their username, date of account creation, the number of videos they have viewed, and the latest date they watched a video.This provides an idea of how many videos were watched at different times of year.

```
FUNCTION get_viewer_info RETURN SYS_REFCURSOR IS
  viewer_cursor SYS_REFCURSOR;
BEGIN
  OPEN viewer_cursor FOR
    SELECT
      v.user_name,
      u.date_created,
      COUNT(v.video_name) AS num_videos_viewed,
      MAX(v.date_watched) AS date_watched
    FROM
      viewer v
      JOIN user u ON v.user_name = u.user_name
    GROUP BY
      v.user_name, u.date_created;

  RETURN viewer_cursor;
END get_viewer_info;
```

Question 4

How many videos were watched during each month of a calendar year? Use the results from Question 3.

This question builds off of the last question by retrieving information about viewers on a monthly basis, including their username, the year-month of video watching, and the number of videos watched in each month. This further allows us to filter how many videos were watched per month across many years.

```
FUNCTION get_viewer_monthly_info RETURN SYS_REFCURSOR IS
  monthly_cursor SYS_REFCURSOR;
BEGIN
  OPEN monthly_cursor FOR
    SELECT
      v.user_name,
      TO_CHAR(v.date_watched, 'YYYY-MM') AS year_month,
      COUNT(v.video_name) AS num_videos_watched
    FROM
      viewer v
      JOIN user u ON v.user_name = u.user_name
    WHERE
      v.date_watched IS NOT NULL
    GROUP BY
      v.user_name, TO_CHAR(v.date_watched, 'YYYY-MM');

  RETURN monthly_cursor;
END get_viewer_monthly_info;
```

Question 5

How many days from the creation of an account was each video watched?

This code will provide a cursor containing information about viewers who watched that video, including their username, the video name, and the number of days since their account was created when they watched the video. This provides an idea of the age of the account versus what videos they watched.

```
FUNCTION get_days_since_creation(video_name_in VARCHAR2) RETURN SYS_REFCURSOR IS
  days_since_creation_cursor SYS_REFCURSOR;
BEGIN
  OPEN days_since_creation_cursor FOR
    SELECT
      v.user_name,
      v.video_name,
      TRUNC(v.date_watched - u.date_created) AS days_since_creation
    FROM
      viewer v
      JOIN user u ON v.user_name = u.user_name
    WHERE
      v.date_watched IS NOT NULL
      AND v.video_name = video_name_in;

  RETURN days_since_creation_cursor;
END get_days_since_creation;
```

Question 6

What is the name and the number of views of the most popular channel? Consider the most popular channel as the channel with the most views.

With this question, we can figure out what is the most popular channel that one of us has ever watched.

```
FUNCTION get_most_popular_channel RETURN SYS_REFCURSOR IS
  popular_channel_cursor SYS_REFCURSOR;
BEGIN
  OPEN popular_channel_cursor FOR
    SELECT
      c.channel_name,
      MAX(c.num_views) AS num_views
    FROM
      channel c;

  RETURN popular_channel_cursor;
END get_most_popular_channel;
```

Question 7

Who watched the most popular channel, when did they watch it, and what videos did they watch? Use the results from Question 6.

This info can be used to see if there are any patterns in the channel's popularity and how long the channel has been popular.

```
FUNCTION get_most_popular_channel_viewers RETURN SYS_REFCURSOR IS
  popular_channel_viewers_cursor SYS_REFCURSOR;
BEGIN
  OPEN popular_channel_viewers_cursor FOR
    SELECT
      v.user_name,
      v.date_watched,
      v.video_name
    FROM
      viewer v
      JOIN video vid ON v.video_name = vid.video_name
      JOIN (
        SELECT
          c.channel_name,
          MAX(c.num_views) AS max_views
        FROM
          channel c
      ) max_channel ON vid.channel_name = max_channel.channel_name
    WHERE
      v.date_watched IS NOT NULL;

  RETURN popular_channel_viewers_cursor;
END get_most_popular_channel_viewers;
```

Question 8

What is the total amount of videos that have been rewatched at least 2 times by each user (Total of at least 3 different views)? Create a procedure to find and display this information.

This can show how many videos each of us choose to rewatch for a variety of reasons.

```
PROCEDURE find_rewatched_videos IS
  rewatched_videos_cursor SYS_REFCURSOR;
BEGIN
  OPEN rewatched_videos_cursor FOR
    SELECT
      v.user_name,
      v.video_name,
      COUNT(*) AS num_views
    FROM
      viewer v
    WHERE
      v.date_watched IS NOT NULL
    GROUP BY
      v.user_name, v.video_name
    HAVING
      COUNT(*) >= 3;

  CLOSE rewatched_videos_cursor;
END find_rewatched_videos;
```

Question 9

What is the average amount of content watched in minutes over a certain span of time? Create a procedure that calculates this number.

This will allow us to find out what our viewing patterns are and if our patterns overlap or correlate with national trends.

```
PROCEDURE calculate_average_watch_time(start_date_in DATE, end_date_in DATE) IS
  average_watch_time_cursor SYS_REFCURSOR;
BEGIN
  OPEN average_watch_time_cursor FOR
    SELECT
      v.user_name,
      AVG(v.length) AS average_watch_time
    FROM
      viewer v
      JOIN video vid ON v.video_name = vid.video_name
    WHERE
      v.date_watched BETWEEN start_date_in AND end_date_in
      AND v.date_watched IS NOT NULL
    GROUP BY
      v.user_name;

  CLOSE average_watch_time_cursor;
END calculate_average_watch_time;
```

Question 10

How many days after a video was posted was that video watched by the user? Create a function that can calculate this amount.

This will allow us to deduce an average of the number of days a user takes to watch a video after it was posted.

```
FUNCTION get_days_after_posted(video_name_in VARCHAR2) RETURN SYS_REFCURSOR IS
  days_after_posted_cursor SYS_REFCURSOR;
BEGIN
  OPEN days_after_posted_cursor FOR
    SELECT
      v.user_name,
      v.video_name,
      TRUNC(v.date_watched - vid.date_posted) AS days_after_posted
    FROM
      viewer v
      JOIN video vid ON v.video_name = vid.video_name
    WHERE
      v.date_watched IS NOT NULL
      AND v.video_name = video_name_in;

  RETURN days_after_posted_cursor;
END get_days_after_posted;
```

Team Descriptions

Dawson Spratley

Both team members have done significant work and have come a long way when considering the workload that was minimized to 3 people being offloaded to just 2. The work was split evenly with Matthew being responsible for questions 7-10 and I being responsible for 1-6. However, given the question we took, the lack of personal data to be analyzed, being down a team member, and the direction we had gone with this project we did not utilize any provided dataset and had provided our own “dummy data " that essentially covered the requirements for the premise of the project. We both feel as though the idea and concepts were learned in an effective manner and intend to utilize the thinking skills and SQL Language in other aspects of future endeavors of our careers.

Matthew Poole

This project has been hectic and unpredictable for our group and I think my partner and I worked very well to overcome the obstacles we faced. After Milestone 1, one of our group members dropped the class, reducing our group from 3 to 2. This loss of a group member made the following Milestones more difficult, but we worked together to finish them on time. For Milestone 2, I made 4 of the questions and Dawson made the other 6, which we thought was a reasonable split. For Milestone 3, the presentation, and the final report, we worked together to solve our created questions, generate “dummy data” in place of the lacking amount of personal data, and finish both the presentation and the final report. We both believe that our work together helped us in implementing the ideas and concepts we learned about PL/SQL. We also believe that this project allowed us to experience hardships while working in a group and how to overcome those hardships by working together.

Resources Cited