Assignment – 4

Project Cont'd

Consider an online video-sharing platform like YouTube which hosts tens of thousands of channels and crores of users.

You have to analyse the data and provide meaningful insights on the type of content that drives engagement, users growth, and many more to all the stakeholders. Let's roll our sleeves up for an insight analysis!

Database

The sample database consists of tables that store the information of users, channels, videos, genres and likes/dislikes.

Note:

channel_user table

channel_id	user_id	subscribed_datetime
100	1	2020-12-10 10:30:45
100	7	2020-10-10 11:30:45
...	...	...

`channel_user` table stores the data of the channel_ids and their subscribers' user_ids.

First row in the table represents that the user with user_id = 1 is subscribed to the channel with channel_id = 100 at `2020-12-10 10:30:45`

user_likes table

user_id	video_id	reaction_type	reacted_at
1	10	LIKE	2020-12-10 10:30:45
7	10	DISLIKE	2020-10-10 11:30:45
...	...	...	...

Similarly, `user_likes` table stores the data of video_id and the user_ids who reacted to the video.

video_genre table

video_id	genre_id
10	201
10	202
...	...

Similarly, `video_genre` table stores the data of video_id and the ids of the genres that the corresponding video belongs to.

Let's dive in to analyze the in and outs of each part of the data. Here we go!

1. Get the top 10 channels for which more number of users are subscribed in the year 2018.

   Note:

   In case, if the no_of_subscribers are same, then sort the output in the ascending order of channel_name.

   Expected Output Format:

channel_id	channel_name	no_of_subscribers
...	...	...

```
SELECT DISTINCT channel.channel_id, channel.name AS channel_name,
count(channel_user.user_id) AS no_of_subscribers FROM channel
 INNER JOIN channel_user ON channel_user.channel_id = channel.channel_id
WHERE cast(strftime('%Y', subscribed_datetime) AS integer) = 2018
GROUP BY channel.channel_id ORDER BY no_of_subscribers DESC, channel_name ASC LIMIT 10
```

2. Get the number of users who positively engaged with at least one video of Disney Channel (channel_id = 352).

   Note:

   - Consider positive engagement as a LIKE for a video uploaded by the Disney channel.

   Expected Output Format:

no_of_users_reached

```
SELECT count(DISTINCT user_likes.user_id) AS no_of_users_reached FROM video
 INNER JOIN user_likes ON user_likes.video_id = video.video_id WHERE video.channel_id = 352
 AND user_likes.reaction_type = 'LIKE'
```

3. Get the number of subscribers for each channel.

Note:

- Sort the output in the descending order of no_of_subscribers, and then in the ascending order of channel_name.
- If there are no subscribers for a channel is 0, then keep the no_of_subscribers as 0.

Expected Output Format:

channel_id	channel_name	no_of_subscribers
...	...	...

```
SELECT
 channel.channel_id,
 channel.name AS channel_name,
 count(channel_user.user_id) AS no_of_subscribers
FROM channel
LEFT JOIN channel_user ON channel_user.channel_id = channel.channel_id
GROUP BY channel_name
ORDER BY no_of_subscribers DESC, channel_name ASC
```

4. Get the number of videos uploaded by the "News for you" channel in the year in 2018.

Expected Output Format

no_of_videos
...

```
SELECT count(*) AS no_of_videos FROM video
INNER JOIN channel ON channel.channel_id = video.channel_id
WHERE channel.name = 'News for you'
AND cast(strftime('%Y', published_datetime) AS integer) = 2018
```

5. Get the number of users subscribed for the "Taylor Swift" channel in every month in the year 2018.

Note:

- Sort the output in the ascending order of month_in_2018.
- Ignore the months that have subscribers_per_month as 0.

Expected Output Format

month_in_2018	subscribers_per_month
1	....
2	....

```
SELECT
 cast(strftime('%m', subscribed_datetime) AS integer) AS month_in_2018, count(*) AS subscribers_per_month
FROM channel_user INNER JOIN channel ON channel.channel_id = channel_user.channel_id
WHERE cast(strftime('%Y', subscribed_datetime) AS integer) = 2018 AND channel.channel_id = 399
GROUP BY month_in_2018 ORDER BY month_in_2018 ASC
```

6. Get the number of videos published by each channel.

Note:

- If a channel did not upload any video, keep the no_of_videos as 0.
- Sort the output in the ascending order of channel_name.

Expected Output Format

channel_name	no_of_videos
...	...

```
SELECT
channel.name AS channel_name, count(video_id) AS no_of_videos
FROM channel
LEFT JOIN video ON video.channel_id = channel.channel_id
GROUP BY channel_name
```

7. Get all the channels that published at least 5 videos in the year 2018.

Note:

- Sort the output in the ascending order of channel_id.

Expected Output Format

channel_id	channel_name	no_of_videos
...	...	...

```
SELECT channel.channel_id, channel.name AS channel_name, count(video.video_id) AS no_of_videos
FROM channel INNER JOIN video ON video.channel_id = channel.channel_id
WHERE strftime('%Y', published_datetime) = '2018'
GROUP BY channel.channel_id HAVING no_of_videos >= 5
ORDER BY channel.channel_id ASC
```

8. How many times each user has engaged with the videos of "News for you" channel (id = 366).

Note:

- Consider engagement as LIKE or DISLIKE for a video uploaded by News for you channel.
- Ignore the users who did not engage with the channel at least once.
- Sort the output in the descending order of no_of_reactions, and then in the ascending order of user_id.

Expected Output Format

user_id	no_of_reactions
...	...

```
SELECT
  user_likes.user_id AS user_id,
  count(reaction_type) AS no_of_reactions
FROM user_likes INNER JOIN video ON video.video_id = user_likes.video_id
WHERE video.channel_id = 366
GROUP BY user_id ORDER BY no_of_reactions DESC, user_id ASC
```

9. Get all the videos that have more than the average number of views.

Note:

- Sort the output in the ascending order of name.

Expected Output Format

name	no_of_views
...	...

SELECT name, no_of_views FROM video

WHERE no_of_views >(SELECT avg(no_of_views) FROM video) ORDER BY name

10. Get all the distinct user_ids who liked at least one video uploaded by Android Authority Channel **(channel__id = 364)** but didn't like the video uploaded by Tech savvy channel with **video_id = 1005.**

Note:

- Consider reaction_type LIKE as liked
- Sort the output in the ascending order of user_id.

Expected Output Format

user_id
...

SELECT DISTINCT user_ID AS user_id

FROM video

LEFT JOIN user_likes ON video.video_id = user_likes.video_id

WHERE video.channel_id = 364 AND user_likes.reaction_type = 'LIKE'

EXCEPT

SELECT DISTINCT user_ID

FROM video

LEFT JOIN user_likes ON video.video_id = user_likes.video_id

WHERE video.video_id = 1005 AND user_likes.reaction_type = 'LIKE'

ORDER BY user_id ASC

11. Get to top 5 viewed videos which belong to both the genres "Action" (genre_id = 201) and "Comedy" (genre_id = 202).

Note:

- In case, if the no_of_views are same, then sort the output in the ascending order of video_name.

Expected Output Format

video_name	no_of_views
...	...

```
SELECT video.name AS video_name, video.no_of_views AS no_of_views FROM video_genre
 INNER JOIN video ON video.video_id = video_genre.video_id
WHERE video_genre.genre_id = 201
INTERSECT
SELECT video.name AS video_name, video.no_of_views AS no_of_views FROM video_genre
 INNER JOIN video ON video.video_id = video_genre.video_id
WHERE video_genre.genre_id = 202 ORDER BY no_of_views DESC, video_name ASC LIMIT 5
```

12. Get the top 5 viewed videos that belong to the "GAMING" genre_type.

Note:

- Sort the output in the descending order of no_of_views, and then in the ascending order of video_name.

Expected Output Format

video_name	no_of_views
...	...

```
SELECT video.name AS video_name, video.no_of_views
FROM video INNER JOIN video_genre ON video_genre.video_id = video.video_id
WHERE video_genre.genre_id = 207
ORDER BY no_of_views DESC, video_name ASC LIMIT 5
```

13. Best time to upload a comedy video:

DunkFest channel is planning to upload a video in the "COMEDY" genre
Give the channel the best suitable hour of the day when users positively engage more with comedy videos.

Note:

- Consider positive engagement as LIKE for the videos in the "COMEDY" genre_type.
- Consider reaction_type LIKE as liked.
- Return the hour in the integer format

Expected Output Format

hour_of_engagement	no_of_likes
5	...

```
SELECT
  cast(strftime("%H", reacted_at) AS integer) AS hour_of_engagement,
  count() AS no_of_likes
FROM user_likes
  INNER JOIN video_genre ON user_likes.video_id = video_genre.video_id
  INNER JOIN video ON video.video_id = video_genre.video_id
  INNER JOIN channel ON channel.channel_id = video.channel_id
WHERE video_genre.genre_id = 202 AND user_likes.reaction_type = 'LIKE'
GROUP BY hour_of_engagement
HAVING hour_of_engagement = 5
```

14. Get active users of the platform. Consider the users who liked at least fifty videos are considered active users.

Note:

- Consider reaction_type LIKE as liked.
- Sort the output in the ascending order of user_id.

Expected Output Format

active_user_id	no_of_likes
...	...

```
SELECT user_likes.user_id AS active_user_id, count(*) AS no_of_likes
FROM user_likes WHERE user_likes.reaction_type = 'LIKE'
GROUP BY active_user_id HAVING no_of_likes >= 50 ORDER BY user_id
```

15. Get all the user_ids who liked at least 5 videos published by "Tedx" channel.

Note:

- Consider reaction_type LIKE as liked.
- Sort the output in the descending order of no_of_likes, and then in the ascending order of active_user_id.

Expected Output Format

active_user_id	no_of_likes
...	...

```
SELECT user_likes.user_id AS active_user_id, count(1) AS no_of_likes FROM user_likes
 INNER JOIN video ON video.video_id = user_likes.video_id
 INNER JOIN channel ON channel.channel_id = video.channel_id
WHERE channel.name = (SELECT channel.name FROM channel WHERE name = 'Tedx')
AND user_likes.reaction_type = 'LIKE'
GROUP BY active_user_id HAVING count(video.video_id) >= 5 ORDER BY no_of_likes DESC, active_user_id ASC
```

16. Get all the potential users. Fetch the user_ids who liked at least 2 videos published by "Disney" channel, and who did not subscribe to the channel (channel_id = 352).

Note:

- Consider reaction_type LIKE as liked.
- Sort the output in the descending order of no_of_likes, and then in the ascending order of potential_user_id.

Expected Output Format

potential_user_id	no_of_likes
...	...

```
SELECT user_likes.user_id AS potential_user_id, count(user_likes.reaction_type) AS no_of_likes FROM user_likes
 INNER JOIN video ON video.video_id = user_likes.video_id
 INNER JOIN channel ON channel.channel_id = video.channel_id
WHERE channel.name = 'Disney' AND user_likes.reaction_type = 'LIKE'
GROUP BY potential_user_id HAVING no_of_likes >= 2 ORDER BY no_of_likes DESC, potential_user_id ASC
```

17. Get top 5 genres in the platform with most positive user activity, i.e., the genres with videos having more number of likes.

Note:

- Consider the reaction_type LIKE as liked.
- If a video belongs to 3 genres, then the likes of the video is counted in all the 3 genres.
- Sort the output in the descending order of no_of_likes, and then in the ascending order of the genre_type.

Expected Output Format

genre_id	genre_type	no_of_likes
...	...	...

```
SELECT genre.genre_id, genre.genre_type, count(*) AS no_of_likes FROM genre
 INNER JOIN video_genre ON video_genre.genre_id = genre.genre_id
 INNER JOIN user_likes ON user_likes.video_id = video_genre.video_id
WHERE user_likes.reaction_type = 'LIKE'
GROUP BY genre.genre_id ORDER BY no_of_likes DESC, genre.genre_type ASC LIMIT 5
```

18. Get the top 3 genre_ids that are liked by the users in India in the year 2018.

Note:

- Consider the name of the country as "INDIA"
- Consider reaction_type LIKE as liked. If a video belongs to 3 genres, then the like is counted in all the 3 genres.
- Sort the output in the descending order of no_of_likes, and then in the ascending order of the genre_id.

Expected Output Format

genre_id	no_of_likes
...	...

```
SELECT genre.genre_id, count(*) AS no_of_likes FROM genre
  INNER JOIN video_genre ON genre.genre_id = video_genre.genre_id
  INNER JOIN user_likes ON user_likes.video_id = video_genre.video_id
  INNER JOIN user ON user.user_id = user_likes.user_id
WHERE user.country = 'INDIA' AND user_likes.reaction_type = 'LIKE' AND strftime('%Y', reacted_at) = '2018'
GROUP BY genre.genre_id ORDER BY no_of_likes DESC, genre.genre_id ASC LIMIT 3
```